THE USENIX ASSOCIATION NEWSLETTER

;login:

JUNE 1995 • VOLUME 20 • NUMBER 3

CONTENTS

FROM THE EDITOR

OPINION

USENIX NEWS

SAGE NEWS

FEATURES

FEATURES (cont.)

STANDARDS REPORTS

BOOK REVIEWS

ANNOUNCEMENTS & CALLS

ETC...

USENIX
The UNIX and Advanced Computing Systems
Professional and Technical Association

USENIX®

;login: is the official newsletter of the USENIX Association.

- *;login:* (ISSN 1044-6397) Volume 20, Number 3 (June 1995) is published bimonthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.
- $24 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $50 per year.
- Second-class postage paid at Berkeley, CA and additional offices.
- POSTMASTER: Send address changes to ;login:, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

EDITORIAL STAFF

- Rob Kolstad, Editor *<kolstad@usenix.org>*
- Tina Darmohray SAGE Editor *<tmd@usenix.org>*
- Nick Stoughton, Standards Report Editor *<nick@usenix.org>*
- Carolyn S. Carr, Managing Editor *<carolyn@usenix.org>*
- Eileen Curtis, Proofreader

MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528-8649
Fax: 510 548-5738
Email: <*office@usenix.org*>

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, and announcements to *;login:*. Send them via email to <*login@usenix.org*> or through the postal system to the Association office. Send SAGE material to <*tmd@usenix.org*>. The Association reserves the right to edit submitted material. Any reproduction of this newsletter in its entirety or in part requires the permission of the Association and the author(s).

;login: is produced with Framemaker 4.0 software provided by Frame Technology and displayed on an NCD X Terminal, donated by Network Computing Devices. The system is served by a Sun SPARCsystem 10 server. Final output is created by a 600 dpi QMS 860 laser printer, donated by Quality Micro Systems. The newsletter is printed on recycled paper.

SAVE THESE DATES!

Upcoming USENIX Events

5th USENIX UNIX Security Symposium

June 5-7, 1995, Marriott Hotel, Salt Lake City, Utah
Sponsored by USENIX, co-sponsored by UniForum, in cooperation with the Computer Emergency Response Team (CERT) and IFIP WG 11.4.
Program Chair: Fred Avolio, Trusted Information Systems, Inc.

USENIX Conference on Object-Oriented Technologies (COOTS)

June 26-29, 1995, Monterey Conference Center, Monterey, California
Program Chair: Vince Russo, Purdue University

Tcl/Tk Workshop '95

July 6-8, 1995, Royal York Hotel, Toronto, Ontario, Canada
Sponsored by Unisys, Inc. and USENIX
Program Chairs: Ben Bederson, University of New Mexico and Will Wilbrink, Unisys, Inc.
If you would like to attend the workshop, please submit a short outline (no more than 1/2 page) describing your reason for attending. Please send via email to *<w95@system9.unisys.com>* by early June.

USENIX Workshop and Tutorials on Electronic Commerce

July 11-14, 1995, Sheraton New York Hotel and Towers, New York City, NY

9th USENIX Systems Administration Conference (LISA '95)

September 18–22, 1995, Monterey Conference Center, Monterey, CA
Co-sponsored by USENIX and SAGE, the System Administrators Guild
Program Chairs: Tina Darmohray, Consultant, and Paul Evans, Synopsys, Inc.
Authors Notified: June 5; Camera-ready Papers Due: August 1, 1995

USENIX 1996 Technical Conference

January 22-26, 1996, San Diego Marriott Hotel & Marina, San Diego, CA
Program Chair: Bob Gray, US WEST
Abstracts Due: July 18,1995; Authors Notified: August 31, 1995; Camera-ready Papers Due: November 14, 1995

2nd USENIX Symposium on Operating Systems Design and Implementation (OSDI)

November 1996, Location to be announced
Program Chairs: Karin Petersen, Xerox PARC; Willy Zwaenepoel, Rice University

USENIX System Administration Conference (LISA '96)

September 30–October 4, 1996, Chicago Marriott, Chicago, Illinois
Program Chairs: Helen Harrison, SAS Institute; Amy Kreiling, University of North Carolina

For more information about USENIX and its events, access the USENIX Resource Center on the World Wide Web. The URL is *http://www.usenix.org*. Or you can send email to our mailserver at: *info@usenix.org*. Your message should contain the line: *send catalog*. A catalog will be returned to you.

Good Ideas

Where do all those good ideas come from? I'm not so sure that all the good ideas come from "think tanks"– those people in Washington huddling deep in the dark recesses of high security buildings where they have great thoughts that are automatically disseminated to and wholeheartedly accepted by the masses.

I have good ideas sometimes. Sometimes I share them; sometimes I don't. I always live in fear that widely publicizing my good ideas will, in fact, reveal them to be bad ideas, once the light of scrutiny and reason is shed upon them. This is the same fear that keeps me from making bold predictions in public (see Cliff Stoll's new book for an example of this style of bold commentary).

I reckon that most people probably have lots of good ideas. To be fair, we all probably have plenty of bad ideas, too. Regrettably, I know that I don't hear enough about everyone else's good or bad ideas. I wish I did.

You would think that all the newsgroups, electronic mailing lists, and conferences would be full of lots of good ideas. In the electronic arena, there appears to be far more complaining and random opinions than constructive schema for forward-looking growth using the wisdom gained from past events. Conferences have a formal track of good ideas that are documented in the proceedings. I'm sure that there must be additional fertile thoughts in the world.

One of USENIX's great strengths is, to paraphrase Mike O'Dell's words, "Moving information from where it is not to where it ought to be." That's really cool! I'm writing this short editorial as a motivator for those of you who have good ideas but don't share them often. Why not try it? Share them with the appropriate mailing list, newsgroup(s), conference(s), or newsletters (like *;login:*). My generation has been told for 30 years that our brain power is one of the most important things for keeping our nation strong. Let's test that theory by sharing the best of our ideas with others who need them.

RK

To the Editor:

Dear Dr. Kolstad:

Almost a year back, I read a note in *;login:* in which you asked people to contribute articles to the publication. Since I was doing considerable writing in my job at the time, I decided to try to contribute something myself. I was able to meet your editorial standards (I think :-)) and shortly thereafter my first article appeared in your esteemed publication. Since then, I have placed an article in every issue.

I did this for a few reasons: to achieve some small amount of notice, to give back to the USENIX community some of what I've gotten, and to develop a new skill that I might deploy commercially at some point. What I did expect was the small number of notes from readers, correcting various points or commenting on some statement I had made. What I did not expect was the note saying, "I like your article, I've put it outside my office door." Or the note saying, "I like this other article, can I use it in my book?" Or the note asking, "Can I use this column in our new Web publication?" I just wanted to thank you for the opportunity to write for an audience, and to let other potential writers out there know what sort of interesting things happen when you start writing.

Scott Hazen Mueller
<scott@zorch.sf-bay.org>

USENIX BOARD OF DIRECTORS

To communicate directly with the USENIX Board of Directors, you may do so by writing to *<board@ usenix.org>*.

President:
- Stephen C. Johnson *<scj@usenix.org>*

Vice President:
- Eric Allman *<eric@usenix.org>*

Secretary:
- Lori Grob *<grob@usenix.org>*

Treasurer:
- Rick Adams *<rick@usenix.org>*

Directors:
- Tom Christiansen *<tchrist@usenix.org>*
- Dan Geer *<geer@usenix.org>*
- Andrew Hume *<andrew@usenix.org>*
- Greg Rose *<ggr@usenix.org>*

Executive Director:
- Ellie Young *<ellie@usenix.org>*

CONFERENCES & SYMPOSIA

- Judith F. DesHarnais
  Conference Coordinator
  USENIX Conference Office
  22672 Lambert Street, Suite 613
  Lake Forest, CA 92630
  Telephone: 714/588-8649
  FAX: 714/588-9706
  Email: *<conference@usenix.org>*

- Peter Mui/Zanna Knight
  Vendor Displays
  Email: *<display@usenix.org>*

- Daniel V. Klein
  Tutorial Coordinator
  Telephone: 412/421-2332
  Email: *<dvk@usenix.org>*

WWW URL: *http://www.usenix.org*

AUTOMATIC INFORMATION SERVER
To receive information electronically about upcoming USENIX symposia & conferences, finger <info@usenix.org> and you will be directed to the catalog which outlines all available information about USENIX services.

SUPPORTING MEMBERS

ANDATACO
Frame Technology Corporation
GraphOn Corporation
Matsushita Electric Industrial Co., Ltd.
Sun Microsystems, Inc., SunSoft Network Products
Tandem Computers, Inc.
UUNET Technologies, Inc.
Sybase, Inc.

The closing dates for the next two issues of *;login:* are June 14 and August 16,1995.

OPINION

Update on Cryptography Export and PGP

by Greg Rose
<Greg_Rose@sydney.sterling.com>
PGP key ID: 09D3E64D 1994/11/30

"Opinion" is an occasional column published in this space. If you feel strongly about a subject that would be of interest to our community, we will air it here. Please send your article to *<login@usenix.org>*.

Last issue I wrote about the background of PGP and the indictment proceedings against Phil Zimmermann. As I promised, I will be updating the status, hopefully with something in every newsletter for the time being. If this article means nothing to you, you might wish to refer to the previous issue. (April 1995, Vol. 20 # 2.)

Phil Zimmermann and his indictment

There is nothing really newsworthy about the indictment proceedings. The U.S. Government investigation continues. Government agents continue to interview witnesses and present evidence to the grand jury. There is no particular deadline for the actual decision to indict.

Phil Zimmermann himself is receiving some recognition though.

He won the Electronic Frontier Foundation's Electronic Pioneer Award at the Computers, Freedom, and Privacy conference in San Francisco in late March. I find it ironic that he received such an award for the same act that may one day put him in the clink.

Personally, I have a disquieting feeling that things like these make Phillip Zimmermann a more enticing target for the government to pursue. On the other hand, maybe he is getting just enough notoriety to make sure that the proceedings will be "squeaky clean," or perhaps even be stopped. Who knows?

Other legal proceedings

I was peripherally aware of other legal proceedings that have relevance to the rules regarding export law and cryptographic software. I've chased up a bit more detail for this article, and as usual discovered that it is more fascinating than I had previously thought.

Moby Crypto is a software product produced by Grady Ward. As I understand it, it is a compendium of software and a "how to" guide for cryptographic applications. In September of 1993 a software publisher, the Austin Code Works of Austin, TX, had a 9:00PM visit by a U.S. Customs agent presenting a subpoena for all their records concerning a product, Moby Crypto, which contained the complete source for PGP 2.3a among five megabytes of other source crypto code and documentation. Even though there were no executables, the fact it was "software" made it a munition under ITAR (International Traffic in Arms Regulations). Austin later got a munitions license and continues to sell Moby Crypto. However no export license has been, or presumably will be, granted. It would appear that this visit was related to the indictment against Phil Zimmermann.

Grady Ward considers it a first amendment issue: Why is cryptographic information allowed in print media but becomes a munition when transcribed to a diskette?

The same question is being asked by others. Bruce Schneier has written *Applied Cryptography*, a book which is acknowledged as an encyclopedia on the topic, with algorithms and examples for most applications of cryptography. About one third of the book is listings of C source code for many of the algorithms it discusses. The book is freely exportable, as confirmed by a specific ruling from the State Department. This ruling was obtained by Phil Karn, formerly of Bellcore, and now with Qualcomm.

Phil Karn subsequently applied for a similar ruling for the floppy disk containing exactly the same files that were published in the book, and which was obtainable from the author. The ruling from the State Department held that the floppy was indeed covered by the prohibitions under ITAR. This request was sent to the "15 Day CJ Request" officers at the Department of State and the National Security Agency. As you have probably guessed, the ruling took much more than 15 days. In fact, it took nearly three months to come back.

How did the floppy differ from the book, in such a way that one was freely exportable while the other was not? The response from the State Department includes the following sentence:

> The text files on the subject disk are not an exact representation of what is found in *Applied Cryptography.* Each source code listing has been partitioned into its own file and has the capability of being easily compiled into an executable subroutine.

Subsequent appeals have, so far, been unsuccessful.

Much of the information above was gleaned from a wonderful WWW resource that I only learned of recently. The reference is to John Gilmore's page. The URL is *http://www.cygnus.com/~gnu/export.html.* John Gilmore is active in the Electronic Frontier Foundation, among other things.

What is USENIX going to do?

In the editing that happened to get the newsletter out two months ago, a couple of the paragraphs got shifted around to make page layout work better, and the net result was that the article appeared to end with Rob Kolstad's editorial comment that other opinions were, of course, welcome. They, of course, still are. However, what I wrote continued on the next page with a section about what USENIX's official position on the whole issue was. Since the article appeared I have had a number of questions about this, and when I eventually got my copy of the newsletter (here in Australia) I finally understood why the readers hadn't read that bit. So here, just this once, I reiterate it.

First, my disclaimer. The article in the last issue, and everything above, is written by me, Greg Rose, and the opinions are mine, and specifically NOT the opinions of the USENIX Association or its Board of Directors.

The USENIX Board of Directors thought fairly hard about whether or not to support Phil's defense. It turns out that we can't, whether we wanted to or not, because it would jeopardize our tax exempt status as a 501(c)3 organization, whose primary purpose is education. It is not clear that a donation of this type falls within the USENIX charter, and the high cost of doing the necessary legal research to investigate other like-cases would be prohibitive. However, keeping the members advised about the status of this issue is certainly educational. To this end USENIX intends to:

- Request a regular piece for *;login*: keeping you advised of the state of affairs. (This is the second such article. The fact that the article happens to be written by a board member is serendipity).

- Get "snitch reports" of the indictment and, if it goes that far, prosecution, along the lines of the ones we do for the POSIX committees; publish these in the newsletter and as press releases.

- Prepare "Friend of the Court" submissions on the subject if that appears relevant.

- Possibly release a position paper to the press.

- Let you know how to donate, if that is something you want to do. (I have. In the last issue I published the boring details. For this article I just ask you to send email to me [*Greg_Rose@sydney.sterling.com*] or Phil Dubois [*dubois@csn.org*], and we can get the details back to you.)

USENIX Member Benefits

As a member of the USENIX Association, you receive the following benefits:

- Free subscription to *;login:*, technical features, system administration tips and techniques, international calendar of events, SAGE News, media reviews, Snitch Reports from the USENIX representative and others on various ANSI, IEEE, and ISO standards efforts, and much more
- Free subscription to *Computing Systems*, the refereed technical quarterly published with The MIT Press
- Access to papers from USENIX Conference and Symposia, starting in 1994, via the USENIX Online Library on the World Wide Web *<http://www.usenix.org>*
- Discounts on registration fees for the annual, multi-topic technical conference, the System Administration conference (LISA), and the various single-topic symposia addressing topics such as object-oriented technologies, security, operating systems, electronic commerce, and mobile computing – as many as seven technical meetings every year
- Discounts on the purchase of proceedings from USENIX conferences and symposia and other technical publications.
- Discounts on BSDI, Inc. products. BSDI information: 800 800-4BSD
- Discounts on the five volume set of 4.4 BSD manuals plus CD published by O'Reilly & Associates and USENIX
- Savings (10-20%) on selected publications from McGraw-Hill, The MIT Press, Prentice Hall, John Wiley & Sons, and O'Reilly and Associates
- Special subscription rates to the periodicals *The LINUX Journal* (Phone: 206 527-3385) and *UniForum Monthly, UniNews*, and the annual *UniForum Open Systems Products Directory*
- Right to join SAGE, the System Administrators Guild

To become a member or receive information regarding your membership status or benefits, please contact *<office@usenix.org>*.
Phone: 510 528-8649.

USENIX NEWS

President's Letter

by Steve Johnson
<scj@usenix.org>

Conforming to UniForum

As USENIX old-timers know, UniForum (formerly /usr/group) was, in effect, split off from USENIX in the early '80s, as the pressure to take UNIX commercial began to grow, and USENIX resisted the commercialization of its bi-annual technical conferences. As such, we have had a close but not always comfortable association with UniForum from its inception. For many years, we were able to meet in the same city but in a different hotel, allowing our attendees to gawk at the exhibits if desired, but keeping our own tone and agenda. Polls of the membership suggested that they liked this strategy quite a lot, and we were encouraged to continue doing this.

Unfortunately, this has proved to be harder than we expected. UniForum had been rotating its conference location on roughly a three year cycle between San Francisco, Dallas, and Washington DC. A couple of years ago, they decided to stop moving around and put their annual conference in San Francisco; scheduling problems dictated a change of time from January to March. The USENIX Board and staff felt that trying to hold our Winter Conference in March and then have a summer conference in June was unworkable, so we broke ranks with UniForum, continued to have our winter conferences in January (in nice, i.e. warm, locations), and have tried to cooperate with UniForum in other ways.

The USENIX staff and I had an intense series of discussions at the latest UniForum in Dallas with both the UniForum Board and the Interface Group, to see if we could arrange to meet with UniForum in San Francisco in 1997. These discussions were amiable and constructive, but ultimately unsuccessful; the USENIX Board of Directors voted not to attempt to move our conference to align with UniForum in 1997.

Regular *;login:* readers will know that putting on our annual conference is quite an exercise in logistics. We need to have space for plenary sessions, invited and contributed talks, the "Guru Is In" sessions, the terminal room, a dozen or so tutorials on two days, lots of BOFS, etc. Ideally, we would like to have everything in the same place (walking a few blocks at night to a convention center for BOFS is pretty unappealing). Our requirements are stiff enough that, for example, in San Francisco there are only two hotels that can accommodate us. We regularly make hotel arrangements five years ahead. Unfortunately, we were booked into San Francisco in 1997 just six weeks ahead of UniForum – even though we were there first, we were in trouble. This led us to explore cooperation.

Ultimately, our biggest concern was the hotel room rates. The logistical problems mentioned earlier mean that most hotels need to devote every square inch of meeting space to USENIX when we descend on them. In practice, if we can't fill the hotel's sleeping rooms as well, the hotel is unhappy because they can't attract other groups to fill these rooms, since we have all the meeting space.

In 1997, we would likely have had to pay room rates in the $150 to $160 range to meet in San Francisco. Since San Francisco has a lot of cheaper hotels, and a lot of our attendees live locally or have friends who do, we would have trouble filling any hotel that we would fit into. This raises the spectre of our having to pay rent on the meeting rooms as well, raising the cost of the conference registration. We concluded that the cost was too high for the benefits. After negotiating with a number of hotels, we are happy to announce that our 1997 conference will be held in Anaheim during January 6-10, 1997. Our 1996 conference will continue to be held as previously planned in San Diego, January 22-26, 1996.

As part of the Board's discussion, we concluded that we would like to migrate our annual conference back into the June slot, rather than the winter slot. To preserve the sanity of our staff, this will probably mean pushing LISA slightly later into the fall, so we are not putting together its program at the same time we are running the annual conference. Watch this space for further details.

Finally, we are actively exploring with UniForum ways in which we can cooperate and support their conference. We have sponsored some tutorials at their last two conferences, and are planning some workshop activities with them. As we have with SANS, ACM, and IEEE, we will continue to cooperate with UniForum whenever it makes sense to do so.

1994 Financial Statements

STATEMENT OF REVENUE AND EXPENSES AND CHANGES IN FUND BALANCE
For the Years Ending November 30, 1994 & 1993

REVENUE	**1994**	**1993**
Membership Dues	$ 381,058	$ 356,051
Product	160,844	139,582
Conferences	2,451,258	2,148,574
SAGE	80,846	45,201
Interest	60,892	55,890
Other	28,426	5,409
Total Revenue	**$ 3,163,324**	**$ 2,750,707**
EXPENSES	**1994**	**1993**
Membership Services/General Admin.	$ 673,469	$ 581,847
Conference	1,896,964	1,402,327
SAGE Expenses	32,187	26,167
Newsletter & Journal	185,409	160,004
Products	57,272	48,271
Projects	85,997	90,426
Depreciation	18,606	22,036
Total Expenses	**$ 2,949,904**	**$ 2,331,078**
Excess Revenue Over Expenses	**213,420**	**419,629**
Restricted Funds Received	**121**	**20,292**
Fund Balance Beginning of Year	**2,024,215**	**1,584,294**
Fund Balance End of Year	**2,237,756**	**2,024,215**
Unrecognized Gain on Securities	**(20,138)**	**45,260**

BALANCE SHEET
As of November 30, 1994 & 1993

	1994	**1993**
ASSETS		
Current Assets		
	875,588	700,536
	72,935	12,232
	112,310	88,498
	30,724	44,679
Total Current Assets	$ 1,091,557	$ 845,945
Fixed Assets		
	$ 1,223,734	$ 1,249,278
	56,104	42,714
Total Fixed Assets	$ 1,279,838	$ 1,291,992
Total Assets	**$ 2,371,395**	**$ 2,137,937**
LIABILITIES & FUND BALANCE		
Current Liabilities		
	$ 93,498	$ 48,237
	60,279	20,225
Total Liabilities	**$ 153,777**	**$ 68,462**
FUND BALANCE	**2,217,618**	**2,069,475**
TOTAL LIABILITIES & FUND BALANCE	**$ 2,371,395**	**$ 2,137,937**

STATEMENT OF CASH FLOWS
For the Years Ending November 30, 1994 & 1993

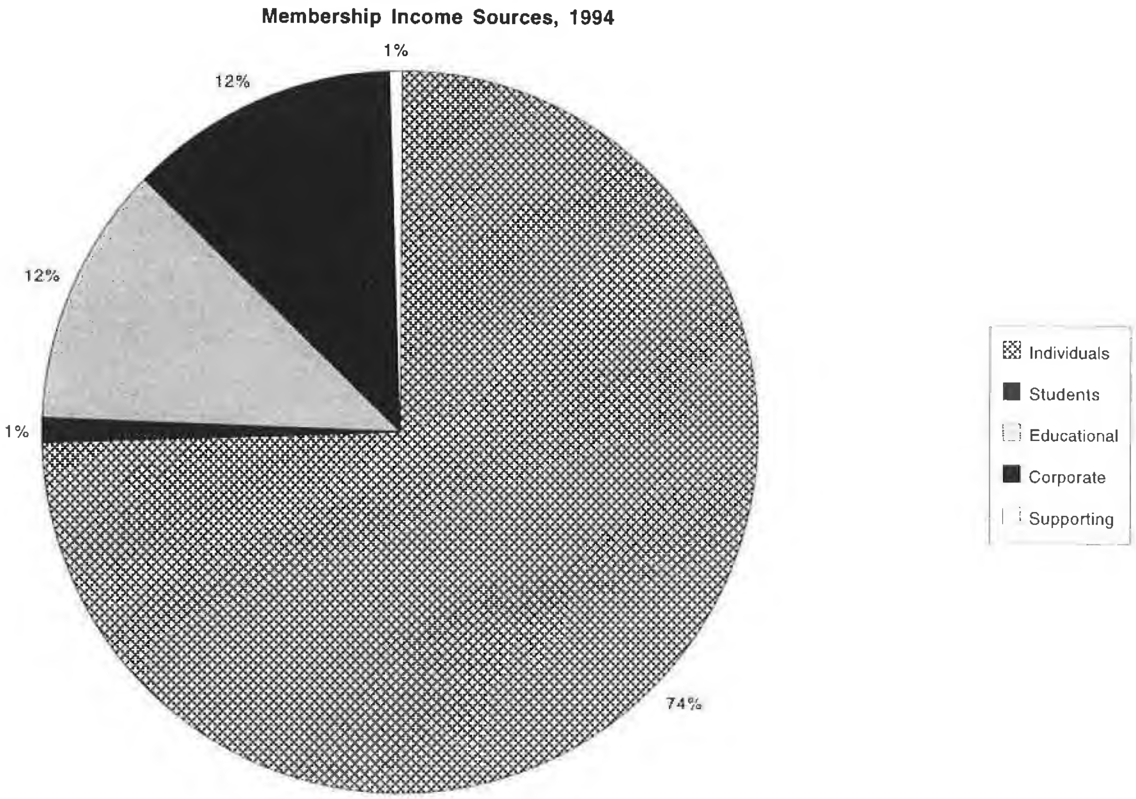
	1994	**1993**
Excess Revenue Over Expenses	**$ 213,420**	**$ 419,629**
Depreciation	18,606	22,036
Decrease/Increase in Receivables	(60,703)	(2,199)
Decrease in Inventory	13,955	70
Decrease/Increase in Prepaid Expenses	(23,812)	(12,768)
Increase/Decrease in Accrued Expenses	45,261	36,829
Increase/Decrease in Deferred Revenue	40,054	(28,660)
Total	**$ 33,361**	**$ 15,308**
Total Net Operating Cash	**$ 246,781**	**$ 434,937**
Receipt of Restricted Funds	$ 121	20,292
Increase in Long Term Securities	$ (39,854)	(318,312)
Decrease in Note Receivable		
Increase of Property & Equipment	(31,996)	(33,383)
Total	**$ (71,729)**	**$ (331,403)**
Net Change in Cash	**$ 175,052**	**$ 103,534**

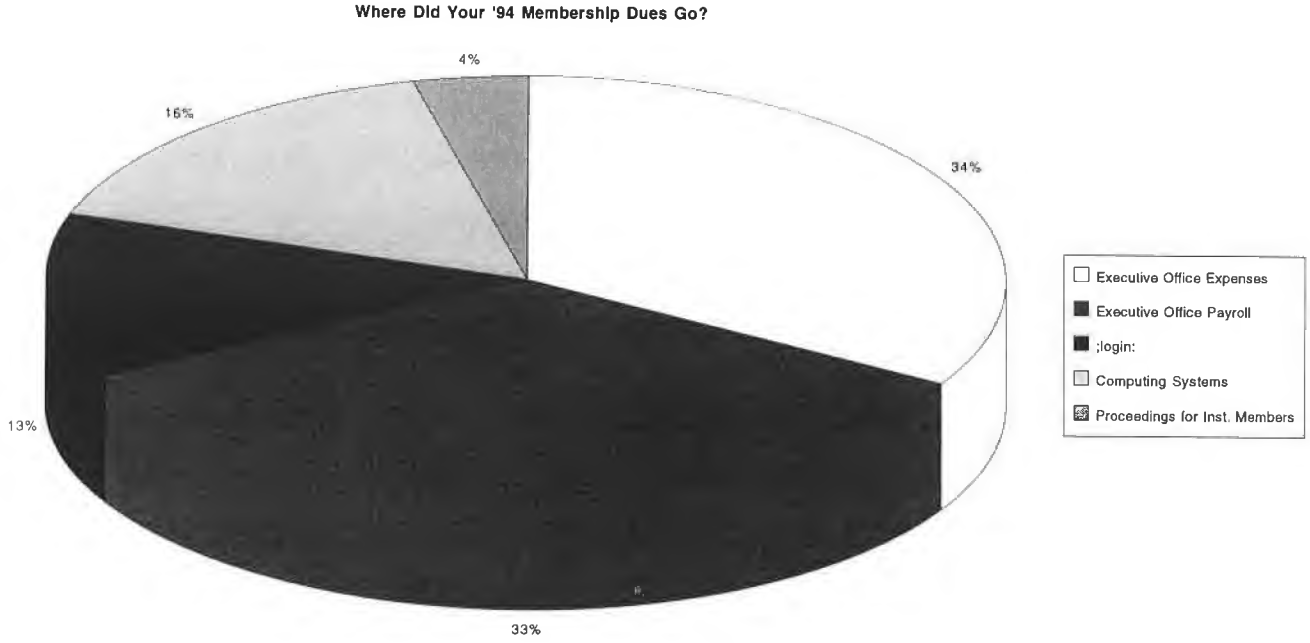
Member Dues

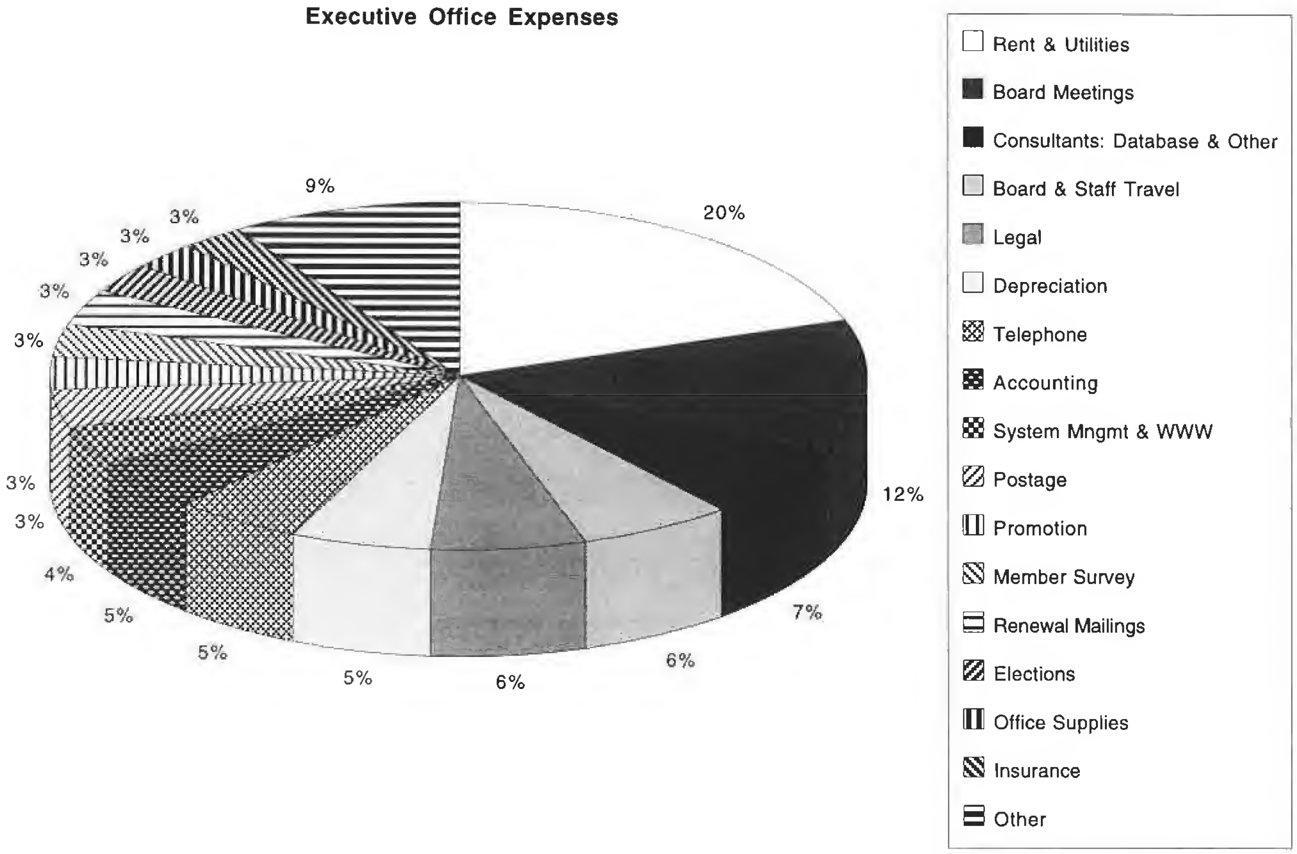
by Ellie Young, Executive Director
<ellie@usenix.org>

Are you ever curious to know how your USENIX and SAGE membership dues are spent? Besides the foregoing financial statements, here are a few charts that might help. The first shows sources of USENIX membership dues income, which totaled $381,000 in 1994. The second chart shows how that money was disbursed. Note that the Conference Office does not receive any monies from membership; it is totally funded by income generated from the conferences and symposia. You should also know that less than one half of the Executive office payroll, general expenses, newsletter and journal expenses are covered by membership dues. The balance is funded by income from conferences and symposia and from publications sales. The third chart shows how the Executive Office spends its money. The "Other" category includes items such as taxes and licenses, bank service charges, and miscellaneous expenses associated with the production of information and materials.

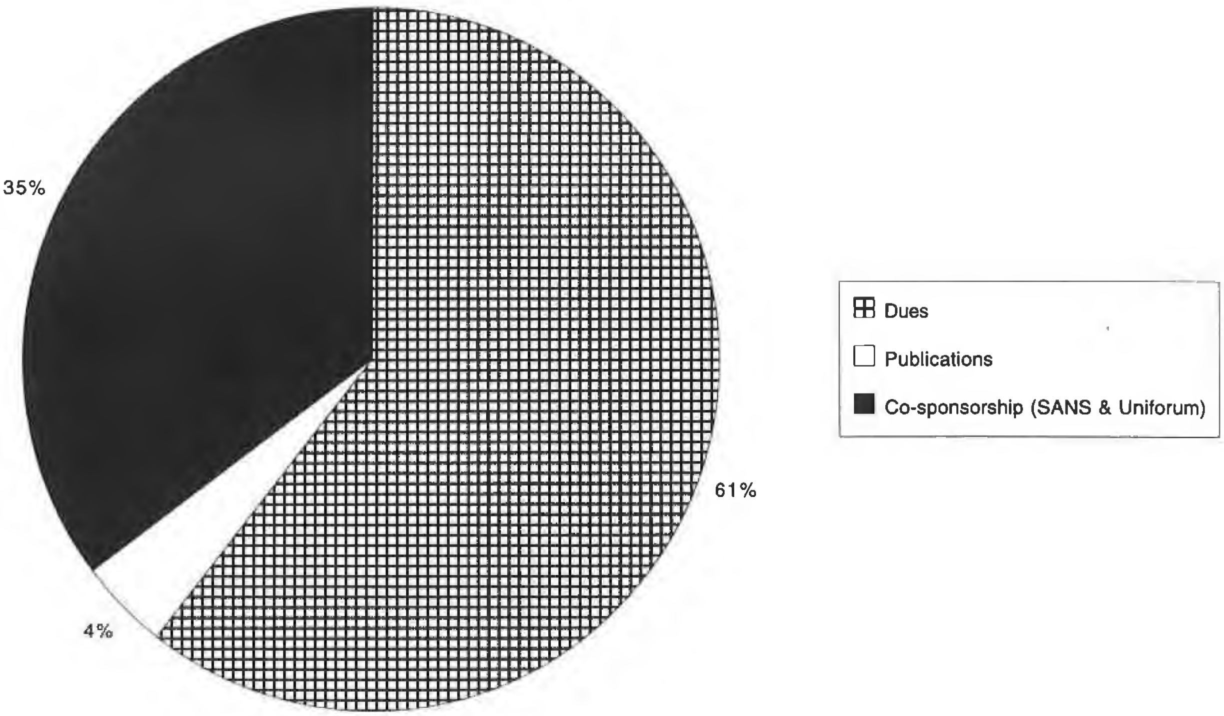
The first SAGE chart shows the sources of all income, which totaled $81,000 in 1994. Two-thirds of the income came from arrangements in which 1) SAGE is an affiliate of the SANS conference which is sponsored by the Open Systems Conference Board, and 2) SAGE shares with USENIX and UniForum the net proceeds from sponsoring a set of seminars organized for the UniForum Conference. The second chart shows how that money was parceled out. In 1994, SAGE income covered 90% of its total expenses (both direct and allocated expenses for administration/payroll). The deficit is covered by funding support from USENIX. Allocated expenses (staff and overhead) are not reflected in the charts and are currently 50% of the entire SAGE budget.



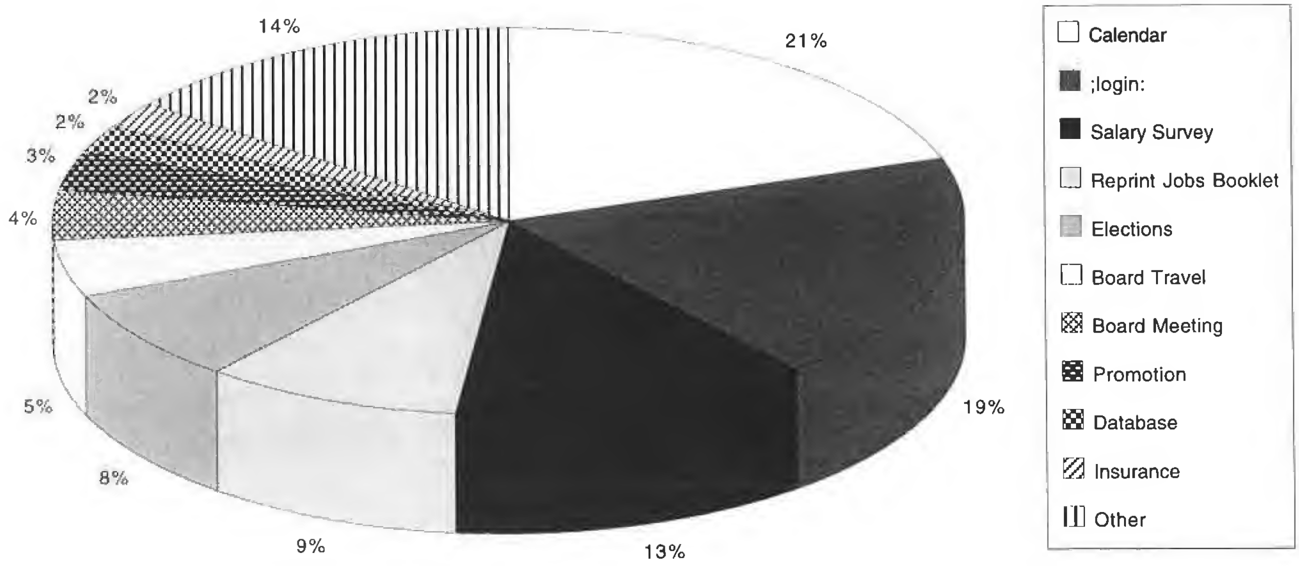

Where Did Your '94 Membership Dues Go?
4%
16%
34%
13%
33%
Executive Office Expenses
Executive Office Payroll
;login:
Computing Systems
Proceedings for Inst. Members


Executive Office Expenses
9%
20%
3%
3%
3%
3%
3%
12%
3%
3%
4%
7%
5%
5%
5%
6%
6%
Rent & Utilities
Board Meetings
Consultants: Database & Other
Board & Staff Travel
Legal
Depreciation
Telephone
Accounting
System Mngmt & WWW
Postage
Promotion
Member Survey
Renewal Mailings
Elections
Office Supplies
Insurance
Other

1994 SAGE Income Sources



1994 SAGE Direct Expenses



Report on the Second USENIX Symposium on Mobile and Location-Independent Computing (MLIC)

by Charles J. Antonelli
<cja@umich.edu>

The second MLIC Symposium was held at the Michigan League on the University of Michigan campus in Ann Arbor on 10-11 April 1995. It drew around 80 participants, and of the 22 papers submitted, 12 were chosen for the technical program.

A well-known bromide around here is "if you don't like the weather, wait five minutes." The Sunday before the workshop we had a splendid snowstorm which closed the Detroit airport long enough to ensure that the pre-workshop reception was an intimate affair indeed.

Monday, April 10

The program chair, Jim Rees, welcomed us to Ann Arbor and went over logistics. He took credit for the weather.

Barry Leiner, Deputy Director of the Computing Systems Technology Office (CSTO) at ARPA gave the keynote address. He spoke of the importance of mobile computing. He viewed the packet radio effort in the 1970s as a precursor to modern mobile systems that did not ultimately succeed because of insufficient hardware.

Barry discussed some of the grand challenge applications defined by the Federal High Performance Computing and Communications (HPCC) Program (more information at *http://www.hpcc.gov/blue95*): digital libraries, crisis and emergency management, electronic commerce, and health care. He selected these applications because they all require extensive mobile computing support. For example, one type of digital library might provide listings of local restaurants, evaluations, and directions to each. This requires bandwidth-adaptive and location-transparent access to permit an "untethered node" to access the library regardless of the communications medium within which the node is immersed, and requires location awareness to direct the user of the node to the chosen restaurant.

Next Barry gave his requirements for mobile information systems. On the hardware side, these included mobile communications technology with a 1 megabit per second (Mbps) throughput at a 1 km range and modular nodes to permit incremental upgrades. Software needs to support mobile hosts as first-class citizens via technologies such as Mobile IP and distributed file systems. Networks need to provide self-configuring, location-adaptive services that provide dynamic effective bandwidth utilization that are compatible with the NII network technology. The end-to-end architecture must support interoperability between networks and quality of service (QoS) management. Finally, mobile applications must be robust in the face of varying bandwidth and sporadic connections.

Randy Katz asked how much of this is a chicken-and-egg problem; the market seems to want voice and low bit-rate messaging services (pagers). Barry replied that he believes the market doesn't lead technology, it seeds it, and if we build the infrastructure the users will come. He pointed out the World Wide Web as an example of this.

After the keynote, the first set of talks addressed connectivity in mobile environments. P. Krishna (Texas A&M University) gave the first presentation, "A Cluster-based Approach for Routing in Ad-hoc Networks." An ad-hoc network is a cooperative engagement of a collection of mobile hosts without a centralized access point, such as a meeting room at a conference or a battlefield environment where it is infeasible to set up a priori static network connections. Characteristic of such environments is a dynamic routing topology in which every host is not in communication range of every other host, so some mobile hosts act as routers.

Krishna said that conventional protocols don't work well in such environments because of the large amount of broadcast traffic necessary to keep routing information consistent. To address this issue, he proposes grouping mobile hosts into clusters and routing between the clusters. A cluster connects to a neighboring cluster via boundary nodes, which are mobile hosts that are members of both clusters. The routing algorithm is based on a standard distance vector protocol driven by tables stored at all hosts. The difference here is that only the boundary nodes must broadcast and rebroadcast routing information as mobile hosts enter and leave the clusters; interior nodes just listen. The fewer the clusters, the more efficient the algorithm; Krishna performed simulations to show that his method of forming clusters always yields two or more nodes per cluster as long as enough nodes are involved in a small enough area. Outside of these limits (e.g., wide area), he said his algorithm is no worse than current protocols.

Krishna was asked why this protocol is better since there seem to be two kinds of broadcasts going on, one among hosts in a cluster as hosts enter and leave, and one among boundary nodes. He replied that while this is true, the interior hosts beacon only, which are short messages. When asked how he maintains a consistent view of cluster membership in the face of communications failures, he said he assumes the wireless link is reliable.

Ajay Bakre (Rutgers University) next spoke about "Handoff and System Support for Indirect TCP/IP". While TCP works well in wired environments, host mobility causes frequent temporary disruptions in connectivity resulting in the loss of TCP segments, which triggers congestion control and seriously limits throughput. The solution described breaks a connection between a fixed host (FH) and a mobile host (MH) into two: a regular TCP connection between the FH and a mobile support router (MSR), and an Indirect TCP (I-TCP) connection between the MSR and the MH. This allows both compatibility with existing networks and the matching of I-TCP to mobile link characteristics.

Ajay's paper focused on the implementation of mechanisms that support I-TCP connections and efficient handoffs of an MH from one MSR to another. I-TCP connections are implemented by having the MSR establish a regular TCP connection with the FH using the IP address and port number of the MH on the MSR side of the FH-MSR connection, and then by establishing a connection between the MSR and the MH using a protocol tailored to wireless environments. When an MH moves to a new MSR, the state held in the old MSR is transparently moved to the new MSR while Mobile IP starts forwarding IP packets destined for MH to the new MSR. The state encompasses a socket pair and any queued data associated with them. The current wireless protocol is TCP modified to reset the transmission timer after a handoff, which causes an immediate slow start instead of exponential backoff.

I-TCP performance measurements showed a slight improvement over wireless TCP when there were no handoffs and a 50% improvement in throughput with handoffs. With large fixed delays inserted into the wireless connection (ten times the normal wireless transmission time), I-TCP showed 100% improvement without handoffs and 300% with handoffs. I-TCP handoff time is dominated by the time necessary to transfer queued socket data; it took 240 ms to hand off with empty socket buffers and 1400 ms with 32 KB of buffered data.

Ajay noted that if an MSR fails silently, the connection fails. So I-TCP introduces an additional point of failure and suggested application-level timeouts since many existing applications do their own end-to-end semantics. When asked why an MH couldn't notice the failure and restart the connection at another MSR transparently, Ajay pointed out that the MH doesn't know the sequence numbers of the MSR-FH connection. More information at *http://paul.rutgers.edu/~acharya/dataman.html.*

"A Wireless Adapter Architecture for Mobile Computing" was presented by John Trotter (AT&T Bell Laboratories). Motivated by the desire for running multimedia applications over indoor wireless networks with renegotiation of data rates, per-connection guarantees and variable bit rates, he described a Flexible Architecture for Wireless Networking (FAWN). The basic hardware element is a PCMCIA wireless ATM two card set with an initial data rate of 625 Kbps (could rise to 20 Mbps). The cards fit into the floppy drive bay of a mobile host. Base stations require one card set per connected mobile host. The FAWN modem card uses frequency-hopping spread-spectrum techniques; bandwidth allocation is controlled by the base station. Most of the time-critical software (MAC, queuing, error detection and forward error correction) runs in the FAWN CPU card, the AAL layer and applications run on the mobile host. When a mobile host detects a reduction of received power from the base, it beacons for a new base and informs software running in the old base.

When asked if the handoffs are reliable, John said this is an important problem and is being actively examined. When asked about range, he said small cells are envisioned, perhaps a couple of offices' worth of distance.

After lunch, the next three talks addressed simulation, emulation, and adaptation of mobile services. Ramki Rajagopalan (Bell Northern Research) spoke about "MCE: An Integrated Mobile Computing Environment and Simulation Testbed." The goal of MCE is to support application development, provide a simulation testbed for evaluation and testing, and provide software support to run applications in actual mobile environments. An objective of MCE is not to require application code to be changed when progressing from simulation to deployment. Applications can be tested by linking with a simulation library, and run in a mobile environment by linking with a mobile environment library with the same API. During simulation, a mobile host is represented by a process, sockets simulate wireless links, and mobility is simulated by checkpointing and process migration.

The MCE routing strategy forwards a message from a mobile host to its mobile support station (MSS), which forwards the message to the destination host's home MSS, which forwards to the mobile's current MSS, which delivers to the mobile host. The MCE handoff strategy is similar to that employed by I-TCP, in that the old MSS forwards state information to the new MSS. When this is simulating with MCE, the process running in the old MSS is stopped, checkpointed, sent to the new MSS, and restarted. The current checkpointing mechanism is simple and does not support file or IPC operations and cannot migrate processes between MSSs of different architectures.

Currently, MCE delivers messages to mobile hosts, performs host registration, unregistration, and location, multicasts messages to sets of hosts, and supports email and a service similar to TFTP.

Nigel Davies (Lancaster University, UK) next spoke on "A Network Emulator to Support the Development of Adaptive Applications." Nigel said the emulator is not a simulator, not particularly accurate (it uses UNIX-based timing), and not scalable (it uses centralized components). It is a tool for developing mobile applications that is simple to use (few changes to applications are required), portable (user-level C code), and easy to configure.

The need to develop software for a mobile application before the associated hardware became available motivated the development of the emulator. The basic approach is to intercept UDP packets traveling between sources and sinks and to introduce a delay to mimic actual network performance. Stubs in the client code communicate with the centralized emulator. Its behavior is specified for known configurations via a configuration file and for unknown configurations by supplying a new send routine and by recompiling the emulator. A typical configuration file entry specifies the throughput of a channel and identifies other channels with which traffic on this channel collides.

Evaluating the performance of the emulator with a pair of processes driving packets at each other, showed a maximum throughput of 4 Kbps with 10-byte packets and 400 Kbps with 1000-byte packets. This was sufficient for Nigel's needs.

A graphical interface for the emulator has also been developed and is freely available. More information at *http://www.comp.lancs.ac.uk/computing/research/mpg/most/emulator.html.*

"A Programming Interface for Application-Aware Adaptation in Mobile Computing" was presented by Brian Noble (Carnegie Mellon University). Brian talked about Odyssey, an API for application-aware adaptation to variations in network conditions. When the network transmission rate drops, say from 2 Mbps to 20 Kbps, running applications must make fundamentally different decisions: a video editor shouldn't degrade quality so it should fetch and store more slowly, while a video player shouldn't slow down playback but rather should degrade the video quality. Brian argued that these different needs cannot be met fully by an operating system but must be specified by the application.

In Odyssey, the system is responsible for tracking changes in the environment by measuring resource availability at the client and representing each as an integer. Applications register tolerances for resources, and the kernel signals an application via an upcall when a resource has strayed outside the specified bounds. (The system is also responsible for adapting to environmental changes in a generic manner for naive applications that do not register.) Registered applications can request changes in system behavior via change requests (equivalent to ioctl calls). The change requests control the fidelity, or the degree to which the data presented match the data stored at the server, in a manner appropriate to the application.

Odyssey defines tomes, which are typed volumes similar to AFS volumes. A tome's type determines resources, associated bounds, and the appropriate fidelity-manipulating operations for all files in the tome. A user-level cache manager located on each client mobile host provides two kinds of services to applications running on the host: the Viceroy is a central point of control that monitors the environment and informs applications of changes, and the Wardens, one per Odyssey type, perform type specific-resource monitoring, provide reasonable default policies for naive applications, and implement fidelity mechanisms appropriate to the type.

Brian has implemented a QuickTime-based video player and Warden. He can select from several video files of the same movie stored on a server at different qualities, based on changes in observed bandwidth. He said a real server would "degrade on the fly," but this would be a greater load on the server. There was some debate with the audience as to whether one number is sufficient to measure the availability of an individual resource. (For example, an application might have frame rate, frame quality, and maximum jitter requirements for a single video stream.)

The last part of the afternoon was reserved for a panel discussion on the topic of mobile routing and networks. Panelists Ajay Bakre (Rutgers), Randy Katz (Berkeley), David Johnson (Carnegie Mellon University), and Ramon Caceres (Bell Labs) were charged with discussing issues such as integrated network vs. separate networks, whether base stations should be bridges or routers, or second-generation concerns such as security, QoS guarantees, and multicasting. The most interesting controversy erupted when Randy Katz put up a slide of a thoroughly wired building with network taps everywhere and said he saw no reason at all for in-building wireless networks, particularly with WaveLan boards costing several times more than Ethernet cards, perceived wireless emissions hazards, and bandwidth issues limiting the number of wireless users in a single room. The ensuing discussion was spirited but friendly.

The USENIX reception was held at the Ann Arbor Hands On Museum, located in a renovated firehouse. The Museum has many interactive exhibits on four floors that encourage children to experiment with basic physical phenomena. Naturally, such encouragement was not needed in this group of adults. In addition to live music and good food, special chairs wired to a synthesizer were provided, so sitting on them in various ways produced sounds more interesting than usual.

Tuesday, April 11

The first event of the morning was a Work-In-Progress session. Randy Katz (Berkeley) spoke about Barwan (Bay Area Research Wireless Access Network) which experiments with overlaying in-building, campus-area, metropolitan-area, and regional area networks and supporting seamless integration across these overlays. He said this is the right approach since there is no single global network architecture and probably never will be.

Patrick Mann (California State University at Fullerton) described his work in integrating a client cache with the OS/2 Installable File System. An LRU cache is maintained on the client and files are fetched from the server on a cache miss. A cache entry is validated by the client before use (NFS model).

Nina Bhatti (University of Arizona) spoke about constructing protocols for mobile computing. Based in part on the Arizona x-kernel, her model defines an experimental testbed that allows plugging in new protocols and strategies and permits their evaluation. In her architecture, a gateway is inserted between the wired (TCP) and mobile side of the network. The gateway talks to a cell manager that handles communications to mobile hosts and implements QoS, acknowledgment, and handoff policies.

Rogelio Valencia (Universidada Publico de Nevarra, Spain) talked about an idea he calls the Mole ("a mouse underground"): a motion detector on the back of a PDA; rolling it around on a desk brings documents into view at the rate at which the PDA is moved, so it appears to scroll over documents physically sitting on the desktop.

Rob Malan (University of Michigan) spoke about work he is starting that aims to avoid network congestion by pipe size estimation. The idea is to maximize throughput by keeping the network "pipe" full, but avoid congestion caused by overfilling. His approach seeks to find and evaluate a bandwidth estimator; the estimated bandwidth and the effective latency will determine the correct pipe size.

Tom Koehler (University of Wurzburg, Germany) spoke about mobile computing in Europe. Of the three technologies in the market (analog cellular, digital cellular, and packet data) he believes digital cellular will win. Some of the problems still to be solved include availability indoors, wide area coverage, software that supports disconnected operation and minimizes network usage, roaming in other networks, and making the technology more understandable to customers. He believes the killer application for mobile computing to be messaging (email and fax).

Thad Sterner (MIT Media Lab), who described himself as one of two cyborgs at MIT, talked about wearable computing. Thad wears a Private Eye display covering his left eye (720x280 pixels), a palm-sized keypad (think of a motorcycle twist grip covered with buttons), and a computer in a shoulder bag. After demonstrating the robustness of his equipment by dropping all of it on the floor, he put it back on to display his notes while speaking. Currently he uses the system as a PDA, but talked about future work in augmenting reality, such as adding a rangefinder to track his fingertip and generating graphics by drawing in the air, adding a GPS for location awareness, adding a camera for aligning repair manual data over a broken copier, or creating a real 3D file system. He would like to look at a whiteboard when reentering a room, and overlay what was written there when he last left, or overlay people's names over their heads while in view. He believes the killer application for mobile computing to be a remembrance agent; while 1% of the CPU is spent on editing, the other 99% ought to be spent searching for similar documents, references and so forth.

The final WIP speaker was Dave Johnson (Carnegie Mellon University) who gave a Mobile-IP Standards progress report. He described the basic routing protocol, in which a mobile host's home agent keeps track of the foreign agent currently bound to the mobile host. The home agent intercepts and tunnels packets to the foreign agent (tunneling conceptually adds an additional IP header), which forwards them to the mobile host. An extension of the router discovery protocol allows mobile hosts to discover new foreign agents. Authentication is done by requiring a security association between a mobile host and a home agent (for example, a shared secret), and MD5 is used to sign messages. Nonces and timestamps are included in messages for replay protection. A problem with the basic protocol is that all packets must pass through the home agent; a solution is to allow a correspondent host to cache the location of the mobile host and do its own tunnelling. Cache consistency is addressed (the home agent always knows the mobile host's correct location) but authentication is hard, potentially requiring all correspondent hosts to have a security association with the home agent. More information at *ftp://ds.internic.net/internet-drafts* (*draft-ietf-mobileip-protocol-09.txt* and *draft-ietf-mobileip-optim-01.txt*).

After the break, the next set of talks addressed disconnected operations. Gabriel Montenegro (Sun Microsystems) first spoke about "System Isolation and Network Fast-Fail Capability in Solaris." Recognizing that UNIX hosts configured for network operation typically freeze when temporarily disconnected, this work defines a fast-fail capability that rapidly informs applications of such disconnections as opposed to waiting for timeouts to expire. A system becomes isolated either by direct user request or when all network interfaces are down. When the system is isolated, an outgoing IP packet triggers an ICMP destination unreachable (source host isolated) error which is propagated to UDP and TCP and returned to the application in one of a few forms, typically ENETDOWN. Naive applications thus fail quickly when the system is isolated. Informed applications and kernel modules can register with a subscription service; connectivity changes trigger an immediate notification. All "standard" network utilities were tested and fast-failed appropriately, sometimes with the error "unknown host" because a name server fast-failed first.

Silvano Maffeis (Cornell University) gave a talk on "A Generic Multicast Transport Service to Support Disconnected Operation." Silvano described an architecture in which multiple clusters, each of which is internally connected by a LAN, are connected to each other via a generic multicast transport service (GTS). A message sent from an application in a cluster is given to a local GTS server; the server then forwards the message to a server in the destination cluster, which delivers the message. The GTS servers tolerate arbitrary communication delays and thus permit the sending of messages to disconnected processes; message persistence is achieved by spooling to a file system. A uniform resource locator scheme is used for addressing messages; a unicast and a multicast protocol (similar to Amoeba's) are supported.

A GTS server is structured in a way similar to the x-kernel and consists of a collection of adaptors organized into a protocol tree. Adaptors isolate tasks such as message passing, message storage, compression, and encryption in replaceable modules. An interface API supports blocking and polled message transmission and reception services. Silvano's group built an application on GTS that supports cooperative software engineering in a disconnected environment; changes made while disconnected are propagated to other group members on reconnection. Distributed copies of directories are kept synchronized via a file replicator similar to rdist; manual intervention is required to resolve conflicts. Other potential applications include a scheme in which paying customers get periodic software updates asynchronously. GTS is publicly available at *ftp://ftp.ifi.unizh.ch/pub/projects/gts/gts*tar.Z.*

Larry Huston (University of Michigan) spoke about "Partially Connected Operation." Larry supports mobile computing at the file system layer by allowing continued access to files cached on the mobile host when the network is unavailable. Similar in spirit to Coda but implemented on AFS, this work modifies file system clients; no server changes are required. In response to a read request issued on a mobile host while disconnected from the network, the cache manager assumes the local copy is coherent and simply returns it to the application. Studies showing that write sharing is rare form the basis for this optimism. Disconnected writes are logged. On reconnection the log is replayed at the server; if there are no conflicts the server copy is updated from the mobile client, otherwise two copies are created and manual resolution is necessary.

A second mode, called fetch-only, maintains cache coherence on reads but keeps the network free of update traffic by logging writes. Most recently, Larry has implemented a partially connected mode that maintains cache coherence on reads and performs log replay in the background. Interactive response time is preserved by performing type-of-service queuing and assigning replay traffic to the lowest priority. Lottery scheduling is employed to prevent starvation and the triggering of congestion control. Larry presented some measurements, one of which showed that certain benchmarks run faster over disconnected SLIP than fully connected Ethernet. Future work includes letting the file system adapt to changing network conditions by switching modes (currently this is a manual operation) and using idle time to heat up a mobile host cache. A member of the audience suggested that it might be important to allow preemption; this would require RPC and server changes.

After lunch, the last group of speakers centered on energy and mobility. Tom Dennehy (an independent consultant) spoke on "Distributed Software Architecture for GPS-Driven Mobile Communications." This architecture, called SANSE, allows mobile information management systems to be built out of modular software parts. One of the design requirements is a redundant user interface that permits voice as well as keyboard input. This redundancy is accomplished through interactors specialized to each function: a voice interactor permits word recognition and perhaps assembly of words into sentences; a screen interactor allows conventional GUI operations; a trap interactor fires in response to elapsed time or distance traveled; and so on. The interactors form part of a software control loop in which events trigger interactors that send commands to processes that need to receive them, the outputs of which feed back to the interactors. Example configurations include a field data collector, in which a low-cost implementation would use only GPS and trap interactors, and a standard mobile host which would use a full complement. SANSE was used to build a portable navigation and geographic information management system that permitted both GUI and voice control of a displayed geographical map.

Monish Gupta (Rutgers University) presented "Energy-Efficient Data Filtering and Communication in Mobile Wireless Computing." This work concentrates on minimizing mobile power consumption by switching off the receiver when not needed, and activating the protocol stack only when necessary, thereby permitting the mobile CPU to enter a low-power "doze mode" state more often. Minimizing the amount of time the receiver is on is accomplished by dividing the transmission stream into a series of alternating short preview and long edition periods. A mobile host (MH) listens during all the preview periods but only to relevant edition periods. A simulation showed that with a preview size of 100 ms and an edition size of 10 seconds, the receiver is off 98% of the time with a delay of 12 seconds.

Activating the protocol stack only when necessary is done via a publisher-subscriber model in which the network interface instead of the mobile CPU does the filtering. The mobile support station (MSS) breaks apart an incoming stream of packets, each of which is destined for some subset of the MHs, into smaller streams with separate multicast addresses based on some prearranged criterion. MHs are informed by the MSS of the appropriate addresses in a short transmission, after which the MH receiver listens for the appropriate packets without involving the MH CPU. Monish gave an example of a stock server in which an MH could use this strategy to listen just for packets about a single stock instead of all stock packets.

Fred Douglis (AT&T Bell Laboratories) gave the final talk of the day on "Adaptive Disk Spin-Down Policies for Mobile Computers." This work seeks to lower the power consumed by a magnetic disk by spinning it down. Spinning it up again consumes energy and time, so the key here is to predict when the disk will be inactive. Traditional methods assign a fixed threshold T and spin down the disk if there has been no activity for that length of time. This performs poorly if the disk is accessed at intervals just longer than T, and in general does not track changing usage patterns well. The solution is to develop an adaptive policy that varies the threshold based on activity. The speaker hypothesized that a user's perceived inconvenience depends on the spin-up delay relative to how long the disk had been idle. Based on this, a function determines if a particular spin-up is inconvenient or not; if so, T is increased, otherwise T is decreased. A trace-driven simulation experimented with various additive and multiplicative factors and came up with several values that decrease the number of inconvenient spin-downs greatly in exchange for only a slight increase in energy.

The last part of the conference was reserved for a panel discussion on the topic of what applications need to know about mobility. Panelists Dick Sillman (Sun Microsystems), Peter Honeyman (University of Michigan), Mahadev Satyanarayanan (Carnegie Mellon University), and Amal Shaheen (IBM) discussed whether or not applications should know anything about mobility, and if so, what should they know and where should they know it. The lively debate on this panel flowed between Satya's argument that the increased dimensions of uncertainty in mobile communications precluded the operating system from hiding mobility, and Peter's position that `/bin/cat` didn't need to know anything about mobility; having to teach any application higher than `/bin/cat` about mobility (e.g., `mpeg_play`) means we have lost the game, and therefore mobility must be hidden by the file system. The panel came to an end with both camps strongly defending their positions.

1995 USENIX Technical Conference Report

Internet at the Turn of the Millennium

by Pavel Curtis, Xerox PARC

Summarized by Jerry Peek
<jerry@ora.com>

Many of the things you do in life are done with other people: browsing at a store, walking in the park, attending a meeting. But when you use the Internet you probably use it alone, one-on-one with a program running on a computer somewhere. Some Internet services are pre-arranged interactions between people – like electronic mail. With a few exceptions, though, visions for the future of the Internet only give more solo services: movies on demand, video conferencing, virtual reality, and so on.

In real life, we do things with other people for fun, collaboration, and learning by watching. Why is it easy to do things together face-to-face – but so hard on the Net? Pavel Curtis and his team at the Xerox PARC's Jupiter Project are working to change that. Their idea is to make "network places." A network place is somewhere that two or more people can share a space together. These places are a lot like physical places. For instance, if you put something down, it'll be there when you come back. (Pavel didn't discuss cyber-thieves. :-))

A place has three characteristics:

1. Co-presence: communication between people, a shared context, and shared access
2. Flexible participation
3. Rich structure

MUDs (Multi-User Dungeons) are one of the few "places" on the Net now. There are hundreds of MUDs: self-contained, isolated worlds that are constructed all the time. MUDs are text-only, though. (If you haven't used a MUD before, FTP to *parcftp.xerox.com* in *pub/MOO/papers* for one of the versions of the file *DIAC92**, "Mudding: Social Phenomena in Text-Based Virtual Realities.")

The MUD named LambdaMOO at Xerox PARC (*lambda.parc.xerox.com*, port 8888) has been running for almost five years. Its programming language, MOO, stands for "MUD, Object-Oriented." It's being used as a basis for more development work, including an internal server that's part of the Jupiter project.

Jupiter has quite a few improvements over traditional MUDs – especially a GUI (graphical user interface), audio and video. The GUI lets MUD objects interact with users through windows that appear on users' screens. When you enter a room, you see small faces of the people in the room and hear peoples' voices. Video can also be used to show presentations, such as the slides and audio from a conference room. A room can access traditional Internet services, too; for example, a library place could let you access online books or a database of geographical information. There can be a whiteboard for sketching. You can leave a written note for other people.

A few technical details: Most processing is done by the clients, not the server, so bandwidth needs for data aren't great. Data connections are Ethernet, T1, and ISDN. The system uses TCP for control connections between a server and its clients. IP multicast connections handle audio and video data between clients. The bandwidth requirements are low enough that many PARC researchers stay connected from their homes, telecommuting via ISDN. The greyscale video isn't real-time, but it's still useful. Jupiter runs on SPARCstation 2 and 10 with no special hardware – PCs and Macintoshes, too. Strong cryptography is used everywhere to help ensure the same confidentiality you'd get in a physical room.

Everything is persistent in these rooms; there's no "save" button to click. The server is programmed by its users with a programming language that's simple, robust, and secure. Lots of people can program at the same time.

The system design is obvious, in many ways, but it raises some tough issues. Your code has to be able to call someone else's code – for instance, to print something for someone else – and running other peoples' code must be completely safe. (As in real life, a few users will be malicious.) You have to be able to change the code while it's running. You can only see and use objects in your current location, and all objects may be shared. But what is an "object?" If you appear to be alone in a room, you MUST be alone. Although most servers can be small – imagine a nationwide conference held in a network place! – it seems to me that scaling could be an issue.

Right now, about 50 people within PARC are using Jupiter. The hope is to make Jupiter freely available for noncommercial use soon.

The team's next project is a bigger web of network places, the Fabric Project. Network places would become part of the network infrastructure. There'd be millions of servers; most clients would also be servers, though some would be shared by many more users. (Think of a server that would handle a convention, like USENIX, via the Net.) All the places and things could be found with unforgeable locators similar to URLs. I can't give exact dates, but the Fabric Project may be starting by the time you read this report. Alpha-level field tests (tens of servers) and a much larger network are planned for the next year or two. The team wants this project to follow the World Wide Web's growth curve. To succeed, the system must be widely adopted and available everywhere.

There are quite a few related papers in the FTP area mentioned above. For example, *NII-Scenarios.txt* describes how Pavel's group believe the National Information Infrastructure should work for people. The *MUDsGrowUp.** files have a paper with some background and details about the projects as they were in 1993; read this paper if you want more than the short summary in this article. If you have any questions, comments, or suggestions about the project's research directions, you should feel free to contact Pavel at *Pavel@PARC.Xerox.Com* or 415 812-4455.

USA Computer Olympiad Sponsors Needed

by Rob Kolstad
<kosltad@usenix.org>

Eleven men and two women have made it to the finals at the USA Computer Olympiad (USACO) which will be held at the University of Wisconsin in Kenosha, Wisconsin from May 31 - June 6, 1995. These students were selected from the list of 265 students from 88 schools who participated in the 1995 Competition Round. In Kenosha, there will be instruction, drills and labs for the winners.

From this group, four finalists will be selected to travel to the world programming championships, the International Olympiad of Informatics (IOI), in the Netherlands from June 25 through June 30. They will be chosen based on programming competence, two timed contests, and general attitude.

Sponsors are needed to offset the remaining $13,000 costs of the three competing rounds and airfares to the championships. If you or your company would like to join USENIX as a sponsor, contact Rob Kolstad at 719 593-9445 or *<kolstad@bsdi.com>*. The money goes entirely to support the program; no fees or honoraria are paid to coaches or sponsors. It's a great program, and a small investment will garner a lot of recognition.

The list of finalists follows:

Name	High School	City & State	Grade	Age
Daniel Adkins	McKinley	Baton Rouge, LA	So	15
Hubert Chen	Upper Dublin	Fort Washington, PA	Sr	18
Russell Cox	Delbarton	New Providence, NJ	Jr	16
Erica Hoffeld	Montgomery Blair	Silver Spring, MD	Jr	16
Yaniv Inbar	New Trier	Wilmette, IL	Sr	17
Amit Khetan	Cranbrook Kingswood	Bloomfield Hills, MI	Sr	17
Travis Kopp	Bear Creek	Littleton, CO	So	15
Patty Lee	Ithaca	Ithaca, NY	Jr	16
Ryan McCorvie	Winter Park	Winter Parl, FL	Sr	18
Preetam Shingavi	Langhan Creek	Houston, TX	Jr	16
Valentin Spitkovsky	Lafayette	Williamsburg, VA	Sr	17
Wesley Tanaka	Punahou	Honolulu, HI	Sr	17
Michael Westover	Thomas Jefferson	Alexandria, VA	Sr	18

In case you are wondering what kinds of problems these students had to solve, two typical ones from the first round of competition follow. They were created (and/or adapted) by Don Piele and the USACO staff:

1. Rational Order

Consider the set of all reduced rational numbers between 0 and 1 inclusive with denominators less than or equal to N.

Here is the set when N = 5:

0/1 1/5 1/4 1/3 2/5 1/2 3/5 2/3 3/4 4/5 1/1

Write a program that, given an integer N between 1 and 100 inclusive, prints the rational numbers between 0 and 1 with denominators less than or equal to N in order of increasing magnitude. Also, count the total number of rational numbers found.

The programs must reject inputs where N is less than 1 or greater than 100.

```
Sample Run
Enter the maximum denominator between 1 and
100 : 5
0/1 1/5 1/4 1/3 2/5 1/2 3/5 2/3 3/4 4/5 1/1
There were 11 rational numbers.
```

Test your program with N=5, 23.

2. Amazing Walks

Given an NxN square divided into N^2 unit cells, it is always possible to walk from the upper left cell, cell 1, to the lower left cell, cell $N^2 - N + 1$, by stepping from one cell to a neighboring cell (one that shares a common side) until you have arrived at your goal having passed through every cell exactly one time. For example, here is a sample Amazing Walk for N=3:

```
----------------
|    |    |    |
| ***********  |
|    |    | *  |
------------*---
|    |    | *  |
| *****   | *  |
| *  | *  | *  |
--*----*----*---
| *  | *  | *  |
| *  | ******  |
|    |    |    |
----------------
```

Write a program to count the total number of Amazing Walks in a square of size N.

```
FOR N=3 THE NUMBER OF Amazing Walks IS 2.
```

Test your program with the following values of N: 2,3,4,5,6.

Note: Without some optimization, your program will take too long for N=6.

Online Library on the Web

Members can now access full papers (ASCII and PostScript) from the 1994-1995 USENIX proceedings on the USENIX Resource Center on the Web: *http://www.usenix.org*. As each future proceedings is published, we will link the abstracts and full papers. If you did not receive your password/membership card via postal mail, please contact <*office@usenix.org*>.

SAGE
THE SYSTEM ADMINISTRATORS GUILD
NEWS

From the SAGE Editor

by Tina Darmohray
<tmd@usenix.org>

I read with interest (as always) the "From the Editor" note that Rob wrote for the last issue of *;login:*. Of course, this time I was looking at it from a different perspective, as I was hoping to get an idea of the kind of thing that I should be writing for the SAGE section.

Rob was talking a little bit about his changing perspective on accomplishments and sustaining success, as he gains professional and life-in-general insight with each passing year.

So, after reading it, I spent some time reflecting on "life," and on how hard it is to sustain a string of successes in personal or professional endeavors. As I thought about it, I realized sustaining performance is difficult in itself, but it also seems that with each additional performance plateau which is reached, the height of the bar that must be cleared to claim that additional "success" is just a little bit higher than before.

All in all, I was thinking that Rob's theory could be applied to SAGE, as an organization. Lots of folks poured immeasurable effort to bring SAGE from an idea to a reality, and then again from reality to a secure existence. Each time, it seems the effort required to get to the next plateau is increased just a little. But, for the system administration profession as a whole, those efforts seem worth it.

With all that in mind, I'll try to continue the quality of the SAGE *;login:* section that Bryan McDonald ably set as the first successful plateau. And, with the system administration community's help, we can strive to clear the next bar. In the coming issues you'll see the addition of some "regular columns" brought to you in the SAGE section. I hope you'll like them. As always, suggestions and contributions are welcome; you can send them to me at *<tmd@usenix.org>*.

Perl Practicum: Fun With Formats

by Hal Pomeranz
<hal@netmarket.com>

Before Perl became a general purpose programming language, it was PERL: the Practical Extraction and Report Language. You can find the evolutionary remains of Perl's humble beginnings hidden away in dark corners of the language. Formats, for example, are a Perl language construct with a syntax unlike any other Perl construct and which generally have functionality that can be emulated with other routines (notably `printf()`). For these and other reasons, most people first learning Perl seem to skip over information about formats, but if you write any reasonable number of scripts to produce reports from long files of data, formats can be a valuable tool.

SAGE, the System Administrators Guild, is dedicated to the advancement of system administration as a profession. Now just two years old, SAGE's membership continues to increase steadily, and there is growing recognition of SAGE as a representative in system administration issues. SAGE brings together system and network administrators for:

- professional and technical development,
- sharing of problems and solutions,
- communicating with users, management, and vendors on system administration topics.

SAGE News Editor

- Tina Darmohray
  <tmd@usenix.org>

SAGE Board of Directors

- Elizabeth Zwicky, President
  <zwicky@usenix.org>
- Hal Miller, Secretary
  <hal@usenix.org>
- Pat Wilson, Treasurer
  <paw@usenix.org>
- Kim Carney
  <kim@usenix.org>
- Paul Evans
  <ple@usenix.org>
- Bryan McDonald
  <bigmac@usenix.org>
- Paul Moriarty
  <pmm@usenix.org>

SAGE Working Groups

Group	Chair
sage-certify	Arch Mott
sage-edu	Paul Evans
sage-ethics	Ed Gould
sage-jobs	Tina Darmohray
sage-locals	Rene Gobeyn
sage-online	Mark Verber
sage-policies	Lee Damon
sage-pubs	Bryan McDonald
sage-security	Arnold & Laura de Leon
sage-stds	Janet Frazier
sage-vendors	Dave England

You can contact these groups via email: *<their-name@usenix.org>* for example, *<sage-certify@usenix.org>*.

SAGE Discussion Groups

sage-managers
sage-outreach
sage-pt

SAGE Online Services

Email server:
majordomo@usenix.org

FTP server:
ftp.sage.usenix.org

WWW URL:
http://www.sage.usenix.org

SAGE Supporting Member

Enterprise Systems Management Corp.

Simple Reporting

One of the first useful Perl applications I wrote was a little program to balance my checkbook: the application reads in a file of data containing all of the transactions I have made to date, and prints a nicely formatted statement with a running balance. I originally wrote the output portion using `printf()` statements, but when I gave the code to Tom Limoncelli, he sent it back to me with all of the `printf()` statements replaced with format code. Darn it, his version was nicer (but my checkbook was balanced first).

I wanted to make the data file as easy to type as possible, so the format is very simple. The first line of the input file is the starting balance, in pennies (no need to type a decimal point and no floating point arithmetic). Each of the following lines represents a transaction: four tab separated fields giving the check number or transaction code, the date, a description, and the amount (again in pennies). Deposits and other credits to the account are represented as negative values (I seem to put money into my accounts much less frequently than I take it out). Figure A, below, shows a simple program to read this input file and generate a statement of the account.

The first four lines in the example are a format declaration. The first line defines the format's name. When the `write()` function is called to print a line of formatted data, it uses the format named for the currently selected file handle. In our example, the program is sending the report to the standard output. Note that if no format name is specified, `STDOUT` is assumed, but it is always better to name formats explicitly, even when you are using `STDOUT`.

The second line is a picture of how each output line will look. Each group of characters beginning with an `@` is an output field specifier – everything else is a literal (e.g., the `$` signs at the beginning of the two money fields). Less-than (`<`) signs mean that the field should be left justified, and greater-than (`>`) signs mean right justified; the pipe symbol (`|`) specifies centered fields. Numeric fields are indicated with hash marks (`#`) and an optional decimal point. The field width is the number of special characters, INCLUDING the `@` sign (in the example below, the first field is six characters wide, the second is five, etc.). This enables the picture to resemble a somewhat abstract but perfectly aligned example of the output.

The picture's third line associates a variable with each field. When the `write()` function is called, the current value of each of the named variables is printed using the specified format. It is clearer to read if you to try and line up the variable specifications with their associated field specifications on the line above.

The last line of a format declaration is always a dot on a line by itself. This terminates the format declaration.

Format declarations can appear anywhere in the program. Figure A contains two format declarations: one before the code and one after. This was done to make the point; in your own code, I recommend you group all formats together near the top of the script. If there are multiple formats with the same name in the program, the one defined last will be the one that gets used.

If a format with the special name `top` is defined in the program, this format will be printed at the beginning of each page of formatted output. The special variable `$=` defines the number of lines per page; 60 is the default, but you can assign a smaller number if you like (for example, when printing to a terminal or small window). The special variable `$-` gives the number of lines left on the current page. You can force a new page by setting `$-` to 0. However: DO NOT mix `print()` and `printf()` statements with `write()` or else the `$-` variable will not be decremented correctly.

Dirty Tricks

While you can define a special `top` format for page headers, there is no way to define a format for page footers. There is, however, a trick for dealing with this situation. While `write()` usually uses the format named for the file handle that the output is going to, you can use a different format by assigning the alternate format's name to the special `$~` variable. The trick then, is to keep track of the number of lines left on the page and emit a special footer format at the bottom of the page. Figure B shows the program logic for doing this.

First we introduce a new `footer` format and a new global constant, `$footer_depth`, which is the number of lines that the footer occupies on the page. The `footer` format in our example uses yet another special variable, `$%`, which gives the current page number (numbered starting with 1).

Each time we emit a line with `write()`, we check `$~` for the number of lines remaining on the page. When we have exactly `$footer_depth` lines left, it is time to write the page footer. To write the footer, we simply set `$~` to the name of the footer format (`footer`, this example), issue a `write()`, and then reset `$~` to the usual format (`STDOUT`) before getting the next line from the transaction file. This line will appear on the next page after the header in the usual fashion.

While this method works very cleanly, when each `write()` statement only outputs a single line, anticipating the end of page when using multi-line formats can get tricky. Also notice that no footer will be output on the last page. Additional code would have to be added after the `while()`loop to output additional blank lines and the footer. This is left as an exercise to the reader.

If you ever want to change header formats for any reason – for example if you wanted a large header on the first page, but only minimal headers on the other pages – you can use the special `$^` variable. This variable behaves like `$~`, but selects the header format instead. Never set `$^` (or `$~` for that matter) to a non-existent format because this will cause your program to exit with a fatal error at run time. If you want a null header, never define the `top` format at all, or set `$^` to an empty format.

Multi-Line Formats

Consider a couple of important facts about the `top` format in the two examples. First, there are no field definitions anywhere in the format declaration. It is perfectly legal to have a format with no field declarations, though in practice you will probably only do this for header formats.

Second, the format declaration defines multiple lines of output. This also is perfectly legal and each line can have zero, one, or more field declarations in it. The general pattern for multi-line format declarations is one line of field descriptions, followed by a line containing the variables associated with those fields, followed by another line of field descriptions, etc.

Figure C shows an interesting use of multi-line formats. For purposes of the example program in Figure C, we are assuming a function called `mailparse()` which processes email messages one at a time from the standard input. For each message, `mailparse()` puts all header information in a global associative array, `%header`, indexed by the header tag (e.g., `From`, `To`) and all of the body lines in a global scalar variable called `$body`. Figure D shows an example of the output from Figure C. By the way, my editor never sent me that message: I made it up. Like all writers, I am always early for all deadlines. Well, that last part was a lie, but I really did make up the email message.

There are a number of new constructs in the `message` format in Figure C. First are the fields that begin with ^ instead of `@`. For these fields, Perl outputs as much text as will fit in the field and then removes that text from the string variable. By stacking several ^ fields together using the same long string, you can output that string as a block of text with a ragged right margin, as shown in figure C with both the body of the message and the `Subj`: line. The special `$:` variable (last special variable in this column, I promise) is the set of characters that on which Perl can legally break the line; the default value for `$:` is `\n -` (newline, space, or hyphen).

The special `~~` marker on the last line means "keep outputting lines until all variables (`$body` and `$header{Subject}` in this case) are exhausted." This is useful for situations where you are not sure how long your text may run, but you want to be able to output all of the information. You can put the `~~` anywhere on the line, but it is best to put it in a very visible location (the beginning of the line is almost always best).

Conclusion

I have run across many Perl programs with complex `printf()` blocks that would have been much easier to write and much more readable if the developer had used formats instead. If you need to quickly produce reports, or output large amounts of tabulated data, formats are an extremely effective tool.

Figure A

```
format STDOUT =
@<<<<< @>>>> @<<<<<<<<<<<<<<<<<<< $@######.## $@######.##
$code, $date,$descript,             $amt,      $balance
.

open(INP, "transactions") || die "Can't read transactions file\n";
chop($penny_balance = <INP>);
while (<INP>) {
   chop;
   ($code, $date, $descript, $penny_amt) = split(/\t/);
   $penny_balance -= $penny_amt;
   $amt = $penny_amt / 100;
   $balance = $penny_balance / 100;
   write;
}
```

```
close(INP);

format top =
.
Trans: Date: Description: Amount: Balance:
====== ===== ============ ======= ========
.
```

Figure B

```
format top =
Trans: Date: Description: Amount: Balance:
====== ===== ============ ======= ========
.
format STDOUT =
@<<<<< @>>>> @<<<<<<<<<<<<<<<<<<< $@######.## $@######.##
$code, $date,$descript,           $amt,       $balance

format footer =

Page @###
   $%
.

$footer_depth = 2;

open(INP, “transactions”) || die “Can’t read transactions file\n”;
chop($penny_balance = <INP>);
while (<INP>) {
   chop;
   ($code, $date, $descript, $penny_amt) = split(/\t/);
   $penny_balance -= $penny_amt;
   $amt = $penny_amt / 100;
   $balance = $penny_balance / 100;
   write;
   if ($- == $footer_depth) {
     $~ = "footer";
     write;
     $~ = "STDOUT";
   }
}
close(INP);
```

Figure C

```
format message =
Date: @<<<<<<<<<<<<<<<<<<<<<<<<< ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
      $header{‘Date’},          $body
From: @<<<<<<<<<<<<<<<<<<<<<<<<< ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
      $header{‘From’},          $body
To  : @<<<<<<<<<<<<<<<<<<<<<<<<< ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
      $header{‘To’},              $body
Subj: ^<<<<<<<<<<<<<<<<<<<<<<<<< ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
      $header{‘Subject’},         $body
~~ ^<<<<<<<<<<<<<<<<<<<<<<<<< ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
    $header{‘Subject’},         $body
.

$~ = "message";
while (<STDIN>) {
   &mailparse();
   write;
}
```

Figure D

```
Date: Tue, 11 Apr 1995 16:39:06      Hal-- What's the status of your Perl
From: tmd@iwi.com (Tina M. Darmo      article for the upcoming issue of ;login;?
To  : hal@netmarket.com (Hal Pom      Rob needs to review the article before
Subj: Your ;login: article is         giving it to Carolyn for typesetting.
      *OVERDUE*                       Please send email soon-- the fate of the
                                      universe is at stake. --Tina
```

Getting To Know SATAN

by Shawn Instenes
<shawni@celene.rain.com>

Some background

SATAN is an acronym for Security Analysis Tool for Auditing Networks. It comes from Dan Farmer *<zen@fish.com>* and Wietse Venema *<wietse@wzv.win.tue.nl>*, two well-known members of the computer security community. It was first discussed in the paper "Improving the Security of Your Site by Breaking Into it" (mostly in its Appendix A). It was released to the general public on April 5, 1995.

SATAN points out possible network vulnerabilities a host or network may have, and it also details assumptions of trust between machines that it discovers. SATAN, as configured "out of the box," can test for these potential problems:

- NFS export to unprivileged programs
- NFS export via one's own portmapper
- NIS password file access
- REXD access
- Sendmail vulnerabilities
- TFTP file access
- remote shell access
- unrestricted NFS export
- unrestricted X server access
- writable FTP home directory

While network scanning tools have been available for some time now, none is as easy to use as SATAN. Through a little PERL magic, SATAN can use any forms-capable WWW browser to provide an elegantly simple user interface. There are help hyperlinks embedded in the vulnerability reports to help you interpret your results, links to programs that can help you correct the problems, and links to papers and reference documents. In all, it's a remarkably useful tool, with plenty of information in a form that is accessible to a system administrator who does not have time to keep up with the security hole of the week.

SATAN's release was quite controversial in the computer security community. It's already prompted the release of two SATAN scan detectors (see Listing One). While these detectors will provide some advance warning of a hostile scan, even from tools other than SATAN, they can be evaded. Your best bet is to secure the problems that these tools discover. Handy tips for doing that is included in the documentation that comes with SATAN itself.

SATAN requires PERL 5 and `fping` to run. SATAN has been tested on a range of UNIX-based operating systems; it's likely to run on yours. A C compiler is used to build a few tools; some porting work may be required if you attempt to run SATAN on obscure platforms. If you want to use SATAN's reporting capability and nifty HTML-based user interface, a forms-capable WWW browser is required. Building SATAN normally requires nothing more than "`reconfig`" and "`make <osname>`".

Pitchfork in hand . . .

Before you use SATAN, *read the documentation.* I cannot stress this enough. If misconfigured, SATAN will merrily follow discovered trust links all over the Net, ringing alarms everywhere. Always limit SATAN to your own network, unless you have permission from a remote network. You can limit SATAN by editing variables in config/satan.cf or through the HTML forms in your WWW browser. Double-check the entries in the form before you click "Start Scan."

When you actually run your first SATAN scan, start small. Treat SATAN like a big dog who will maul anyone in the way, and put it on a short leash. Try it out on a single host. Watch what ends up in the log files on that host. You might be surprised by how little is logged; this really depends on your vendor and local customization.

After you have a feel for what it will do for one machine, try a subnet. Now you can see trust information that SATAN identifies between hosts. With this information, SATAN tries to work out the most "interesting" hosts, from an intruder's point of view: these hosts are likely trusted by one or more other computers. Depending on your network, it may be worthwhile to pay special attention to these hosts when securing them, patching them, or just watching their log files. Revealing this web of trust is something new in network scanning programs. SATAN uses a configurable set of rules to infer possible new targets to scan based on trust information it discovers. It can uncover trust extended to hosts not under your control, which other scanners (directed only at a single host or network) often miss.

Adapting SATAN

So, now that you've unpacked SATAN and fixed the problems that you discovered, your network is secured. No more problems. Right?

Well, I doubt it. A sad fact of computing today is that most operating systems rely on large, complicated programs to make security decisions for us. Sometimes it's not easy to tell if a program is making those decisions. Or doing them correctly. Thus, the *new* security problems.

SATAN can be easily extended to search for newly discovered problems. If you can write a program that tests for a vulnerability, then you can write a SATAN module to report it, and let SATAN produce the neatly-formatted reports about your network. Don't be surprised if new SATAN modules are posted to Usenet newsgroups or mailing lists.

In Listing Two, there is a SATAN module that is designed to report SMTP server version numbers that are reported in the banner (in the case of Sendmail, this is set in the sendmail.cf file, using "De"). Just to make things interesting, it will report an occurrence of 8.6.11 as a security problem that leads to a root shell. (I do not know of any such vulnerability, at the time of this writing.) Listing Three contains a small change to make to the file rules/services that is distributed with SATAN, in order to make the SATAN reports look better without writing much more new PERL code.

To use this module, place a new entry (shown in Listing Four) in the "@light" array in config/satan.cf, and then place the smtp.satan file in the bin directory. Make sure that "tcpscan.satan" is not in the "@light" array, or remove "smtp" from tcpscan's default argument list; this will prevent the reports from getting messy later, as we're using the "smtp" service name differently than tcpscan does.

Run SATAN, and select a "light" attack level so that the modules listed in "@light" are invoked on your targets. Once SATAN finishes collecting the data, you can view the summary of SMTP server versions on your net by selecting "SATAN Reporting and Analisys" and then "By Class of Service" from the SATAN start page. The SMTP versions are listed individually. You can also search by individual host names; SATAN will have "SMTP (version) server" listed in the results for each host that runs SMTP.

The program is straightforward. SATAN tools are invoked with the fully qualified domain name of the current target. After some setup, the program establishes a socket to the SMTP port, fetches one line of text, and then issues the SMTP command QUIT to disconnect in an orderly fashion. The line of text is broken into a hostname and banner information, which is written as a SATAN database record using utility routines available in perl/misc.pl. If the target host does not respond to the connect or the connection times out, then a record indicating the target is not running a SMTP server is written. SATAN collects the standard output of the tool program and continues with the next target, until there is no more work to do.

With a little imagination, you can cause SATAN to report all kinds of things about your network. Want to know which hosts are running the network-based time-management software you just received an update for? Which have the old version of the operating system? With a SATAN module, you can find out quickly.

Listing One: URLs of note

SATAN: *ftp://ftp.win.tue.nl/pub/security/satan-1.0.tar.Z*

Satan requires: PERL 5: *ftp://prep.ai.mit.edu/pub/gnu/perl5.001.tar.gz*

"Improving the Security of Your Site by Breaking Into It": *ftp://ftp.win.tue.nl/pub/security/admin-guide-to-cracking.Z*

Courtney is a PERL script that, in combination with tcpdump, recognizes and logs telltale signatures of scanning tools like SATAN:

Courtney: *ftp://ciac.llnl.gov/pub/ciac/sectools/unix/courtney.tar.Z*

Courtney requires: PERL 5 (see above) tcpdump: *ftp://ftp.ee.lbl.gov/tcpdump-3.0.tar.Z*, also get: *libpcap: ftp://ftp.ee.lbl.gov/libpcap-0.0.tar.Z*

GABRIEL is a standalone C program that also recognizes SATAN scan signatures: *ftp.lat.com:/gabriel-1.0.tar.Z*

Author's test SATAN module: *smtp.satan: ftp://qiclab.scn.rain.com/pub/security/checkers/smtp.satan*

Listing Two: smtp.satan

Install this in $SATAN_HOME/bin/stmp.satan, where $SATAN_HOME is the directory SATAN is installed in.

```
#!/usr/local/bin/perl5.000
#
# This is a sample SATAN module, for
# demonstration purposes. It doesn't do
# anything useful.
```

```
#
# report SMTP server banners to SATAN. I'm
# assuming an occurrence of "8.6.11" in the
# SMTP banner indicates a problem that
# can give you a root shell. (Not true as of #
5-Apr-1995 to the best of my knowledge.)
#
# 5-Apr-1995 Shawn Instenes
#

require 5;
use Socket;

#
# None of the SATAN routines should go to
# test mode.
#

$running_under_satan = 1;

#
# I want &fix_hostname and &satan_print from
# SATAN itself.
#

require 'perl/fix_hostname.pl';
require 'perl/misc.pl';

$waittime = 6;

#
# seconds to wait before timeout
#

$target = shift;

die "No target!\n" unless $target;

#
# We'll set an alarm so that if the connect()
# or banner read blocks, we'll continue
# anyway. SATAN will kill this process in a
# few seconds anyway, this is just to make
# certain we terminate in a reasonable
# amount of time.
#

$SIG{"ALRM"} = 'timedout';
alarm $waittime;

# a good deal of this is right out of the
# Camel book. #

$sockaddr = "S n a4 x8";
chop ($hostname = 'hostname');
($name, $aliases, $proto) =
getprotobyname('tcp');
($name, $aliases, $type, $len, $thisaddr) =
gethostbyname ($hostname);
($name, $aliases, $type, $len, $thataddr) =
gethostbyname ($target);
$this = pack ($sockaddr, &AF_INET, 0,
$thisaddr);
$that = pack ($sockaddr, &AF_INET, 25,
$thataddr);
socket (S, &PF_INET, &SOCK_STREAM, $proto) ||
die "socket: $!";
bind (S, $this) || die "bind: $!";
connect (S, $that) || &timedout;

#
# Fetch the banner line from the server.
#

select (S); $| = 1; select (stdout);
$_ = <S>; print S "QUIT\r\n";
close (S);

alarm 0;

#
# The following regexp is simple and unlikely
# to suit all environments. Salt to taste.
#

($host, $ver) = /^\d+.(\S+)\s+(\S+\s+\S+)/i;

#
# Emit the vulnerability record. The format
# is documented in the SATAN documentation.
# We're using SATAN's own routine
# &satan_print() for this. All we have to do
# is set variables.
#

$fqdn = &fix_hostname($target, $host);
$host = $fqdn;
$service = "smtp";
$status = "a";

#
# Severity 'rs' indicates this problem grants
# a root shell to an intruder.
#

if ($ver =~ /8.6.11/) {
$severity = "rs";
$trusted = "ANY\@ANY";
} else {
$severity = "";
$trusted = "";
}

$trustee = "";
$service_output = "";
$text = "$ver";

&satan_print();
exit 0;
```

```
#
# Emit a record for hosts that do not respond
# to the SMTP port.
#

sub timedout {
alarm 0;
$fqdn = &fix_hostname($target,
$target);
$host = $fqdn;
$service = "smtp";
$status = "u";
$severity = "";
$trustee = "";
$trusted = "";
$service_output = "";
$text = "No SMTP";
&satan_print();
exit 0;
}
```

Listing Three: Additional line to $SATAN_HOME/rules/services

Add this to the SERVERS section:

```
$service eq "smtp" SMTP ($text)
```

Listing Four: Additional line to $SATAN_HOME/config/satan.cf

Add this to one of the scan level arrays, such as @light:

```
'smtp.satan',
```

So What's This Tickle Stuff All About?

by John E. Schimmel
<jes@sgi.com>

If you hang out with the systems administration community these days you probably hear a lot of buzz about *Tcl*, commonly pronounced tickle. Or, even worse, tickle-teekay for *Tcl* and the *Tk* toolkit. *Tcl* is short for the Tool Command Language, and is really just a very simple interpreted language designed as a library of reusable C routines.

Tcl only has two data types: the string and the hash array of strings. It is relatively inefficient. Code is constantly being reinterpreted each time a routine is called; strings are always being reconverted into numbers or lists. All in all it does not seem to have the attributes of a popular language. So why is it that *Tcl* has become so popular?

As for any good question, there are several answers to *Tcl*'s popularity. The first and primary reason is *Tk*, the "teekay" part of "tickle-teekay." *Tk* is a simple X windows toolkit for *Tcl*. It is completely free and sits on top of the free part of X from the X Windows Consortium. *Tk* mimics the popular look of Motif, but it does not use any of the Motif libraries. And, in many ways improves upon the mimicked widgets.

A second reason is the embeddedness of *Tcl*. It is relatively trivial to add a *Tcl* interpreter to any project you are working on, and it gives the power-users a common programming environment for modifying the tools they use. Instead of hacking together some simple command interpreter each time a new tool is written, now you just include the *Tcl* library, and perhaps add a couple of new commands specific to your tool.

That brings us to the third reason for the popularity of *Tcl*. It is utterly trivial to expand. All of the commands in *Tcl* are basically just C procedures. All commands are called with the same arguments, a pointer to the interpreter state, an argv/argc array, and a static pointer that you can do with as you please. There are common procedures to parse the arguments and handle errors, but for the most part it is as if you built your own standalone program in C, and as such it can do anything that can be done in C. *Tcl* commands can be added and deleted while running, so it is a natural abuser of shared libraries. Whenever you need a command that does not exist, you can go out and look for the command in an index, and load it.

A final reason for the popularity of *Tcl* is the wealth of work already implemented and freely available. Virtually everything you need in order to produce a powerful tool is available on the Internet. There are *Tcl* extensions for SNMP, for GL or Motif windows programming, and for writing powerful chat scripts around interactive programs. There are extensions supporting all of the major databases sold for UNIX. There are numerous widgets available for *Tk* and Motif. And, of course, there are already wrappers around all the standard UNIX libraries.

Tcl was written by John Ousterhout and some students at the University of California at Berkeley in the late 1980s because he found that he was creating a new command interpreter for each of the applications that he was working on, and none of them was really powerful enough to do anything interesting. By creating an interpreter library he was able to shorten the design cycle for individual tools, and provide a common interface into each of them.

At the 1990 Winter USENIX Technical Conference, Ousterhout presented a paper on *Tcl*. He then made it freely available to the UNIX community. Within months there were dozens of hackers fiddling around with it.

By the next USENIX conference there were already many extremely useful applications written using *Tcl* as their base.

Expect was delivered to the USENIX community by Don Libes at the 1991 Summer USENIX Technical Conference. It enables you to write chat scripts around interactive programs that were difficult to automate in shell scripts before.

Tk, also by Ousterhout, premiered at the following Winter Conference. It began as an attempt to create something akin to the HyperCard product from Apple, but ended up as something much more general purpose and simple.

Today, *Tcl/Tk* have an enormous following, and as with any product that gains popularity quickly, it also has a large number of contesters. A couple incidents have arisen in the recent past to cause some fear among the user community. A couple of these are statements from the Free Software Foundation outlining the weaknesses in the base language for *Tcl*; statements from the Perl community regarding the performance of *Tcl*; and, Ousterhout's moving out of the academic community.

The standard answer to performance problems in *Tcl* has always been that it is easy to fall back into C. All of the complicated work is performed by well optimized libraries of compiled code, and only the glue work should be done inside of the wrapper script. *Tk* is a good example of this philosophy. All of the routines to manipulate data on the screen are handled by the standard X library or the base *Tk* library routines. But creating the widget hierarchy, setting resources to change how things look or how they are laid out, and the callbacks, are done in the script. It is relatively simple in a *Tcl* script to change where a button is placed or what color it defaults to. To do this in C would mean recompiling, linking, etc. In *Tcl* all of this can be done on a live application while it is running.

Similarly, the answer to the lack of complex data structures in *Tcl* was that interesting data could be maintained in C structures and manipulated by the commands without it being visible within the script. Again *Tk* is a good example of this. Under the wrapper there is an enormous tree of complex widget structures that gets passed around and manipulated by each of the *Tk* commands. All of this data is invisible to the shell programmer, and there is no need to convert it into strings to pass it around.

Depending on the application, falling back into C for everything you do may not be a reasonable answer. For the standard systems administrator this may be a little more than what they want to do. This is one of the primary reasons for the *Tk*Perl and Scheme*Tk* work that is going on. But, at the present, *Tcl* covers a large subset of the current problem set, it is available and free.

The final cause for fear in the use of *Tcl* was that of Ousterhout's current position, and what that might mean to the language. This is perhaps the most frightening of the problems around *Tcl,* but in reality it is the least interesting. If for some obscure reason Sun Microsystems would privatize the work on *Tcl,* there is enough of a following in the free world that *Tcl* would splinter at the last free release, and live on without them. John's move to Sun has probably done just the opposite. *Tcl* will be better funded, and, being a product of a systems vendor, it will force a response by the alternative companies. Unfortunately, however, supporting *Tcl* as a product will make major changes to the syntax and the like nearly impossible. Companies don't like customer complaints and major changes always generate myriads of complaints.

So, *Tcl* is a popular tool, and perhaps a very useful one for now. Someday there will be a better one, and we will all want to switch over and rewrite all of our applications to use it instead. But for now hack and enjoy. There is nothing quite so pleasing as sitting down for the afternoon and adding a beautiful GUI to your favorite old shell hack.

Watch this location for interesting *Tcl* examples through the coming months. For more information you can read *Tcl and the Tk Toolkit* by John Ousterhout (Addison-Wesley Publishing Company, ISBN 0-201-63337-X), or *Exploring Expect* by Don Libes (O'Reilly and Associates, Inc., ISBN 1-56592-090-2). Interesting USENIX papers are "*Tcl*: An Embeddable Command Language", and "An X11 Toolkit Based on the *Tcl* Language" by John Ousterhout, "*Tcl* and *Tk*: Tools for the System Administrator" by Brad Morrison and Karl Lehenbauer, and numerous papers on *expect* including: "Using *expect* to Automate System Administration Tasks", "*expect*: Curing Those Uncontrollable Fits of Interaction", "Using *expect* to Automate System Administration Tasks", and "*expect*: Scripts for Controlling Interactive Processes", all by Don Libes. USENIX papers are available through the USENIX office, and many of them are available online through the USENIX Web library: *http://usenix.org/publications/library/index.html.*

10 Questions for Sysadmins

by Pat Wilson
<paw@usenix.org>

One occupational hazard most, if not all, sysadmins share is becoming so wrapped up in day-to-day crises that we don't often sit back and view the Big Picture. Here are 10 questions about your job, your contacts, and your habits to ask yourself occasionally (while waiting for the *next* crisis). If your answers to all of these questions are an unqualified "yes," you get a gold star.

1. If I get run over by a truck today, are all procedures documented such that the installation would keep running in my absence?
2. How recently have I monitored the security of my machines? Is the appropriate level of paranoia (trust) being maintained?
3. Do I have a clear conduit to and good communication with my major vendor reps?
4. Do I subscribe to the appropriate mailing lists, read newsgroups, or at least somehow stay in touch with information about platforms/applications from non-vendor sources?
5. Do I belong to a user group, national association or attend conferences where I can talk to others in the same line of work?
6. Do I use good ergonomic techniques on the job (sitting, typing, lifting, etc)? Am I familiar with the warning signs of RSI? If I'm already affected, have I taken steps to minimize further damage?
7. How's my mental health? Do I have good strategies in place to cope with job stress and burnout?
8. Is my relationship with my supervisor good? Are reporting lines clear, and are reviews performed on time?
9. Do I have career plans, and am I on track? Am I in touch with office dynamic/politics that affect me?
10. Is my resume/CV reasonably up to date?

Another Maxim of System Administration

by Paul Evans
<ple@synopsys.com>

Francesco Guiciardini (1483-1540), political advisor to the Medici popes Leo X and Clement VII, and a friend of Machiavelli, is one of my favorite authors. He kept a notebook in which he recorded maxims for political and diplomatic action, which he called "ricordi." [*Maxims and Reflections of a Renaissance Statesman*, translated by Mario Domandi. New York: Harper & Row, 1965.]

Steve Simmons and Elizabeth Zwicky have collected and presented what might be called the ricordi of system administration (see "Simmons' Laws of System Administration," *;login:* July/August 1992), and I've contributed a law of my own in these pages (see "More Laws of System Administration," *;login:* May/June 1993). Here's another contribution to the body of maxims or ricordi of system administration. System administrators seem to have the tendency to use almost any justification to give a solution a foot in the door when they want that particular solution badly enough. They develop justifications for spending money on a particular solution or technology that they think will appeal to those controlling their organization's budget. I have learned never to push a technology, product, or service as a solution for any problem other than the problem I'm actually interested in solving. My experience has been that the ostensible use will completely displace the actual intended use, and the problem you wanted to solve will remain unsolved. Here's how I learned my lesson:

In 1985, I worked for a company that ran 4.2 BSD UNIX on VAX-11/750s for general-purpose computing, had no UNIX workstations, and had no intention of getting one. The company was in the instrumentation business, and it was therefore involved in both hardware and software development. It was not a progressive or forward-looking company – I vividly remember the VP of Engineering telling my manager and me that "UNIX is a false standard of technical excellence being shoved down our throats by the universities." (This statement was made in 1988, long after it was clear that, like it or not, UNIX was not going to just go away.) The system administration group in which I worked made a tremendous effort to get the engineering management interested in the then new Sun workstations as a development platform. The management was at the time only vaguely aware that anyone was using the VAXes for anything other that email and troff, so getting them interested in a better UNIX platform was very difficult.

The manager of the tech pubs department, however, was an aspiring empire-builder and got interested in buying Suns, not from Sun but from Interleaf, as a platform for running the Interleaf office publishing software. We (the system administration group) eagerly cooperated with this effort on the theory that once the machines were in the door and their capabilities became known, they would quickly be taken over by engineering.

In fact, this delayed the introduction of UNIX workstations into engineering use by two years. The workstations arrived with an Interleaf, rather than a Sun logo pasted on the monitor, and the user population invariably referred to the systems as "the Interleafs." Most users, and all of the senior management, were under the misapprehension that the systems were manufactured by Interleaf, and were quite shocked to learn, several years down the line, that they were in fact Sun workstations. The false justification for buying the workstations (office publishing) displaced the real justification (software development) to the great harm of the company.

The second case in which I experienced this phenomenon, centered on Internet connectivity. At this time, I was working with a different company, much more technically astute than my previous employer had been, but, because it was a startup, penny-pinching in the extreme. My boss and I, who agreed on almost nothing else, were determined to get an Internet connection. Our view was that this was just part of the price of playing in the big leagues, which this company aspired to do. This argument went nowhere with the executive staff. A bit later though, the bid for an Internet connection was revived in a different guise. The proposal was to put one of the company's machines out on the net for potential customers (defined very loosely) to play with, and that this would lead to sales. This argument was much more pleasing to the exec staff, with the result that the company's Internet connection was placed in the hands of the Marketing Department. What was supposed to be the company's firewall was, in effect, a public-access machine.

System administrators are in the business of solving problems. My experience has been that being able to solve a particular problem sometimes depends as much on how you justify your proposed solution, as on the rightness of the solution you propose.

Do You Want to Exhibit at LISA '95?

Are you or your company interested in showing your products at LISA 95? Participating in the Vendor Display is a great way to showcase your products to LISA's experienced and influential attendees, and we make exhibiting easy by including everything you need to exhibit with the booth. Contact Peter Mui at the USENIX office for more information:
Phone: 510 528-8649
Fax: 548-5738
Email: <*pmui@usenix.org.*>

1995 SAGE Outstanding Achievement Award

SAGE is soliciting nominations for its third annual Outstanding Achievement Award, to be presented this September in Monterey at the LISA '95 Conference. The SAGE Board of Directors has appointed a special committee to select this year's recipient, and we're inviting your suggestions. The award will go to someone whose professional contributions to the system administration community over a number of years merit special recognition. In 1993, the first recipients of the award were Max Vasilatos and Rob Kolstad, for their role in organizing the early LISA conferences, as well as their general contributions to the system administration community. Last year's recipient was Larry Wall, for his work on Perl and other system administration tools.

The awards committee would like to keep the selection process informal; there isn't a formal nominating procedure, and we will consider all suggestions submitted. So please send in suggestions for people whose professional accomplishments you believe deserve the recognition of a SAGE Achievement Award to *sage-award@usenix.org* by June 16.

Errata

The word "co-workers" was changed to "coriander" in Tina Darmohray's article on page 28 in the April 1995 issue of *;login:*. All editors and typesetter are puzzled and unanimously agree it's the work of Framemaker poltergeists. Perhaps it should have read "a carpool of basil."

The errors on the cover, however, were grand oversights on the part of humans. Apologies to those misnamed:

Exploring Expect was *written* by Don Libes not reviewed by him. The reviewer was Glenn Vanderburg.

Curt Schimmel was the author not reviewer of the book: *UNIX Systems for Modern Architectures, Symmetric Multiprocessing, and Caching for Kernel Programmers.* Thanks to George Neville-Neil for the review.

The X Station: A High-end Workstation in Disguise

From the International Hewlett-Packard Users Conference Maastricht, The Netherlands, 10-14 April 1994

by Ronald van Driel and Daan Reuhman
<driel@prl.philips.nl> <reuhman@prl.philips.nl>

This article describes a computing infrastructure which supports the CAD electronics community at Philips Research, The Netherlands. Rather than implementing a traditional single-person workstation scheme, a computing infrastructure based on X stations has been implemented in which all computing is performed by shared, task-specific server computers. The article starts with some arguments against the use of personal workstations, followed by a discussion on the global architecture of the X station environment including a closer look at the network design. The conclusion shows a fully functional installation servicing over 200 users and suggests cost savings up to 50% when compared to a typical workstation alternative.

Introduction

"When people are limited to the CPU power on their desks, the entire set of computer hardware must be replaced frequently to keep in step with technology."

Plan 9 from Bell Labs, Rob Pike and Ken Thompson, 1988

The CAD support department at Philips Research is responsible for the computing facilities used by the CAD electronics community. Over the last five years the computing function has been implemented as a 400-seat Apollo workstation network augmented with multi-vendor server platforms. Faced with an inescapable transition, it seemed an ideal opportunity to reconsider the suitability of the "one person-one workstation" model of computing in the light of experience.

Technical computing environments have traditionally been constructed out of UNIX-based personal workstations and a few server computers connected by a local area network. Typically, these environments suffer from the following characteristics:

- expensive to keep up-to-date with rapid workstation developments,
- fragmentation of potentially huge amounts of computing resources, and
- widely varying patterns of compute power usage by individual users.

With this knowledge in mind, it seems odd to install dedicated computing devices with a fixed capacity on the user's desk. The workstations are probably as often under-powered as they are over-powered, depending on the job at hand. More effective computing paradigms have already been proposed by two influential research projects: Plan 9 from AT&T Bell Laboratories and Professor Tanenbaum's Amoeba development. It's in the spirit of these projects that we decided to design a new computing infrastructure.

Architecture

At the heart of the architecture is the idea to move the CPU and disk away from the desk to a central pool of computing resources, leaving only a graphical terminal, i.e., an X station, in the office. An X station is made available to each user, while all computing is performed on a shared pool of high-end workstations configured as servers with lots of memory and disk space. Past experience with time-sharing computing (VAX/VMS) strongly suggests avoiding the mix of interactive and batch type computing tasks on the same server. Consequently, the following server classes have been defined:

- login servers,
- compute servers,
- batch servers, and, of course,
- file servers.

Login servers are used to boot the X stations and to run the login sessions. Generally speaking, they deal mainly with the desktop tool set including electronic mail, text editors, word processors, and calendars. The function of the compute servers is to run large interactive applications that require resources beyond the scope of the login servers. Batch servers run the traditional CAD applications for analysis, simulation, and verification which exclusively run in non-interactive mode. The sole purpose of the file servers is to store all user data and application software.

Network

It can hardly be over-stressed that communication in the projected environment is a key issue. Two main types of network traffic are distinguished: the X protocol between the X stations and the compute or login servers, and NFS file traffic between the batch, compute, login and file servers. It is generally advised that the larger NFS packets must not be mixed with smaller X packets on the same network, because heavy NFS traffic may prevent the X packets from entering the network for considerable periods of time. Interactive response will then suffer dramatically.

Looking at the implementation of the network, two practical alternatives come to mind: the common 10 Mbit/s Ethernet and the 100 Mbit/s Fiber Distributed Data Interface, FDDI. The implemented network topology is shown here.

Only 25 to 30 X stations are connected to the same Ethernet segment. Ethernet-level traffic separation is realized by the multiport bridge. The selected bridge is a Cisco AGS+ with an internal bandwidth of over 500 MByte/s.

Status

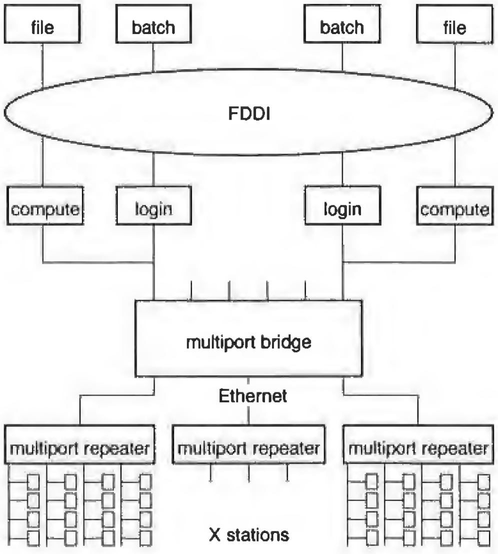
The present configuration comprises 10 login, 12 compute, 6 batch, and 4 file HP 735 servers. File servers have 80 MB internal memory, and store over 20 GB each. The other servers have 272 MB internal memory, and store no user or application data. Some statistics:

HP servers installed	32
HP X stations installed	200
Apollos as X stations	50
Concurrent users	> 200
Users per login server	20--25
Users per compute server	8–12
Total batch users	> 350

Conclusion

The principles underlying the X station infrastructure are to provide stable technology on the desktop and to deliver compute power as a public utility. Sharing is the name of the game. The environment features computing resources on request, support for large jobs, central administration, and focused upgrades. For these reasons alone the X station variant is our preferred environment, even if it would carry the same costs as a workstation alternative. In addition, the efficiency that comes with the principle of sharing resources can lead to savings up to 50% in investment and maintenance.

Finally: the environment is not yet complete, but the introduction is well under way. The experiences so far have proven favorable enough to continue the expansion towards an expected total of 400 X stations.



Object Oriented?

by Scott Hazen Mueller
<scott@zorch.sf-bay.org>

"Object Oriented" (OO) has become such a buzzword that many computer people have taken to ignoring it. Now, I don't look at things from a programmer's perspective, but I do want to know: what will objects do for (or to) me?

What is OO?

For those who have managed to stay unfamiliar with the term, OO is basically a different way of looking at programming. The information space is populated with objects, data structures of a sort, each of which possesses associated functions, or methods. One of the key features is that the objects in OO programming can be associated with real-world objects in a 1:1 correspondence.

One issue that arises frequently is the distinction between what programmers mean when they say "object oriented" and what marketing folks mean. Because real-world objects can be mapped to programming objects, certain user interfaces that display real world objects are described as object oriented, even though the underlying program may not actually be object oriented. This confusion is understandable, but it does complicate the use of the term in its original sense.

One of the main values of OO programming is that objects are supposed to be opaque and reusable. If an object's specification matches your needs, you should be able to use it without worrying about the actual implementation. Assuming that the objects are fully debugged (!), then the technique should make for better code.

OO User Interfaces

Object Oriented user interfaces are a convenient and natural way to deal with actual physical objects. One of the main uses for these interfaces is as an intuitive front-end for system and network management applications. Systems are reasonably well-understood discrete objects, and when they are installed on a network, the object within the application can represent the actual state of the system.

An associated UI technique is the "drag and drop" notion, wherein objects are placed onto (into, in a virtual sense) each other in order to represent some action. Dropping a file onto a representation of a printer is a classic example of this technique. The user interface benefits greatly from the use of real OO programming techniques in the underlying application, because OO includes strong typing, which makes it easier to avoid unwise drops, such as dropping an application binary onto the printer.

Distributed Objects

Because the typical office computer environment is networked to one degree or another, the consortia that are setting standards for OO programming are very interested in making objects network-compatible. This makes a rather large amount of sense, both from the business standpoint of trying to meet the future, and from the technical sense of how objects are implemented.

A trend that is likely to converge with the object efforts is the rise of multiprocessing in the commercial marketplace. Most high-end and mid-range systems vendors have sold multiprocessor systems for years, but now multiprocessing (with identical processors; non-identical multiprocessing has been around longer) is starting to appear in low-end PC servers. The high-end PC technologies of 10 years ago (for example, 80386 systems) are now near the bottom of the heap in offices, and well within reach for many home computer users. If this goes on, most new computers will be multiprocessing systems.

Right now, many multiprocessing systems don't look much like networks. There is one exception. From the standpoint of system architecture, objects look rather like "messages," and loosely-coupled multiprocessing designs send messages between processors. This implies that the work being done to make objects network friendly can be easily leveraged to run on multiprocessing systems if the systems use message passing, implying loose coupling.

By itself, this isn't a very strong argument for making future systems loosely coupled, but loose coupling wins in two other areas. Current tightly-coupled multiprocessing systems suffer from a diminishing returns problem; each processor added to the complex adds less net compute power than the previous processor. Loosely-coupled systems, on the other hand, have been shown to scale linearly to much larger configurations. This does assume that the job mix is not inherently single-threaded, but then object-oriented systems are indeed not single-threaded. The second design win is that loose coupling offers a higher degree of fault resilience; indeed the most-commercially-successful fault tolerant systems (from Tandem Computers, Inc.) are based on a loosely-coupled design.

Another angle on this is that as networks become faster, they will begin to more closely resemble interconnects within a single system. That is, a network of systems will look more like a single loosely-coupled system than has been true in the past. Transitively, a loosely-coupled system could be made to look like a somewhat faster network. Therefore, either arrangement would be completely transparent to the objects, and ultimately to the users. The main difference would be one of speed.

The Bottom Line

Right now, with current OS technology, it doesn't seem useful to support multiprocessing on the desktop. In the future, with object-based OS support, and object-oriented applications, it will become fruitful to move to this sort of architecture. Converging system and network technology with this programming technology will enable new design models that will greatly expand the power available to computer users.

Making a Living in Technical Sales Support

by Rick Umali
<rgu@world.std.com>

Introduction

This article is less about technology than it is about a people-role in technology. Last year, I became a technical sales support person, which is a job that's decidedly less technical than the jobs held by most of the writers of this newsletter.

How it got to be that way

My current position evolved over three years at a small computer company in Cambridge, MA. Small companies have the unique feature that every person usually does "two jobs:" sales people field calls during lunch; programmers handle "phone duty" (our euphemism for technical support).

Whenever the need arose, I made time to help the sales people. I figured it was a good way to help the company's bottom line, and I really enjoyed helping people understand the technology that they were selling (Motif widgets and interface design tools).

I got a special kick when a sales person understood a techie concept, or made a sale because I got on the phone and explained something to a customer. I thought it was a unique challenge to come up with analogies, or to help persuade a customer to buy our products.

In January of 1994, I decided to make a break from programming, to pursue this role of sales support full time. I suppose it's easier to say that I was between projects, but to be more truthful, I wanted to continue being there for our sales department. I felt that every sales department in every technical company should have someone like me to help.

What a day is like

So the burning question now must be: "What do I do?"

I spend a lot of time on the phone, preparing to be on the phone, and decompressing after phone calls. I talk to sales people regularly, via email and in the halls. I occasionally go on customer visits. I make custom distributions of our software on media (a primitive form of just-in-time manufacturing), and make sure that shipping ships it. In short, I do a lot of short jobs.

Soothing customers during "evaluation" phases

Probably the most important aspect of my job is pre-sales support. Pre-sales involves persuasion, phone savvy, and quick thinking.

Prospective customers are in a very open mode: they want information, they divulge requirements, they reveal limitations in their current technical staff. My job is to answer their questions positively, and more importantly, to make a good impression.

Sales is often about impressions. By providing good support before the purchase, a sales person can make a better claim about support after the purchase. By being up-front now about product limitations, I establish a candor that customers truly appreciate.

I've spent a good number of calls telling people that our products cannot help. I've spent time doing off-the-cuff consulting on some of their tasks. I help customers figure out if our company can help them.

Helping sales people understand

It's the common lament that sales people promise things that aren't true, but I find that they promise things that they don't understand. By helping them learn about our technology, they can better understand a person's needs, and wants. I help our sales people understand what our product actually does.

Using Mosaic, I created an on-line system where each of our products had a "Tip Sheet." These Tip Sheets are very much like accessing the FAQ list at *http://www.cis.ohio-state.edu/hypertext/faq/usenet/FAQ-List.html.*

Each tip sheet lists platform availability, current version, common questions and answers, and a list of major functionality. It's an electronic version of me. I'm "growing" the list of major functionality as sales people ask questions. It's a poor man's "knowledge tree" or "expert system."

Of course, in order for me to learn about this technology, I need to communicate with developers as well. Here is where a developer's background helps. I can speak with them on a technical level and send this information back to sales. I usually wade the waters between sales and developers.

What it takes

By the time you reach this far into the article, you're probably saying, "Yeah. . . that's me. I want to be a technical sales person." Here's what it takes.

It takes a different "mentality." A thread on *comp.software-eng* debated the differences between people in support versus people in development. David Brooks wrote "support requires a process orientation, where you get your satisfaction from turning the crank and converting an incoming call to a closed call." This process orientation is at odds with the "abstract orientation" (or "product orientation") most programmers need to be able to program effectively.

Be prepared for slightly different compensation at some companies. *Reseller Management* magazine (November 1994) says at some companies, formal sales engineers provide technical backup, but are compensated differently from sales people. It says that at one company, sales engineers are paid 80% base salary, 20% commission (as opposed to true sales people, who are usually paid 40% of their salary by commission).

Finally, be prepared to learn techie things on your own. It's so much easier to lose your touch in sales. I answer the same set of questions over and over. In three years, these questions won't be interesting anymore. Thus, I've embraced opportunities to learn technology, either on my own time, or by assisting with Q&A or technical support.

Becoming a sales engineer is a perfect way to have a "people job" in the computer world. It's a different way of "being technical" and supporting customers.

On Glass Houses and Thrown Power Bricks or: A Solaris Hater Eats His Hat

Opinion by Lee Damon
<nomad@castle.org>

I left last January USENIX's KOTF BOF feeling very depressed about the future of the individual development of UNIX. During the course of the meeting, our discussion rapidly changed from "what's wrong with this variant of UNIX" to "what can we do about all these well oiled market departments that cause standards to be created, standards that are beyond bad."

I've never been so distressed and depressed about the future and trends in our business. By the time we were done talking, I was ready to consider a career change. Anyone need a mediocre chef?

Then I started thinking. I mean *really* thinking. The problem isn't so much what's wrong with MS Windows, or Solaris, or AIX, or (name your favorite OS to hate), but with the way decisions are made by the companies that produce them. Instead of concentrating on what's bad with these products, we need to encourage the good.

Remember when we used to laugh at the mainframe boys in their Glass Houses? We knew their days were numbered, and we were going to take over. I've got bad news for you: We May Be Next. If we maintain the UNIX-is-the-only-solution-you-will-ever-need attitude, we will find ourselves following them down the same slope to destruction.

Let's not make the same mistake our compatriots in the glass houses made. We need to learn NT, DOS, Windows and Finder. We need to embrace the good ideas we find there, and to help them remove the bad ones. Instead of insulting them with names like Windoz and NoT, let's see what we can learn from, and do with, them.

Today, PCs outnumber UNIX workstations by even larger percentages than workstations outnumber mainframes. You find them everywhere in the office and in many homes. People are familiar with them, and feel they don't need a priesthood to help them use the PC. PCs might not be as functional as our workstations, but most users don't need much beyond what a basic PC can do.

We can not afford to treat PCs and PC Operating Systems as "the enemy" any more. We have a choice: we can fight the reality, or we can cooperate and integrate them into our lives and world views. Think back. When was the last time something really new and exciting that absolutely required UNIX was announced? We need the challenge, we're getting stale, folks.

This point has recently been driven home to me. I've begun evaluating CDE, and immediately noticed a very disturbing resemblance to MS Windows. The reason? So many people are so used to the PC interface that they are demanding it on their other platforms. The result? Standards are being written based on this particular "look and feel."

Microsoft might not write very good software, but they are sure good at selling it. Instead of trying to ignore them, it's time we tried to work with them to improve the usability and reliability of their code. Since we're going to get stuck using it either way, we might as well try to make it better

We have learned much as an industry. We all have many valuable skills. I challenge you to find newer and better ways to use those skills. What else can you do to improve the entire industry? Where can we go next?

We need to be able to change and grow with the times. My greatest fear in all of this is that the future will see "one Operating System for the masses." Whether that Operating System is Solaris or NT or BSD/OS, it would still spell disaster. Variety is critical to the continued growth and development of our industry.

We can't let the companies with strong marketing departments control the industry. We have to stand up to the Suns and Microsofts. We have to tell them we want more than one answer. For that matter, we need more than two or even five answers. We can't let any Operating System "win." We need them all.

Don't let the name get in the way, it doesn't matter whether we call it UNIX, DOS, BOB, even Farfencomputen. We shouldn't care as long as the job gets done. We must keep learning and growing. No one can stop that, and no one should try.

Your challenge, should you choose to accept it, is to learn something good about the Operating System you love most to hate. Then take that good thing and make it work for you. It's time for us to lose the "We are the one true way" attitude.

Everything You've Ever Wanted to Know About X/Open UNIX, and the Single UNIX Specification

by Stephen R. Walli, SRW Software <stephe@usenix.org>

[Author's note: Much of this material first appeared in an article for Open Systems Networking & Computing, *(March 1995), an English technology analysis monthly. It has been reworked to remove much of the historical UNIX material, and edited to be more useful to USENIX's technical audience.]*

In October 1994, X/Open rolled out the Single UNIX Specification, and followed this up with the X/Open UNIX brand program in March 1995. UNIX is now "a trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd."

So what does this mean?

This article provides a brief history of Spec 1170, the work that lead to the Single UNIX Specification, describes the Single UNIX Specification and its context within the X/Open Common Application Environment, and outlines how X/Open brands work in general with specific reference to X/Open UNIX.

POSIX.1, XPG4, and Spec 1170 in Context

A little history is in order. A Common Open Systems Environment (COSE) project was announced in Spring 1993 to develop a common API specification to support application source-code portability. The five primary sponsors were Hewlett-Packard, IBM, Novell/USL, the Open Software Foundation, and Sun Microsystems. However, the project was endorsed by a total of 50 system vendors and 20 independent software vendors (ISV). The intent was to develop a specification of the common interfaces used in traditional UNIX systems, which when implemented by the supporting system vendors, would allow software vendors with applications developed for traditional UNIX systems to more easily port their source-code to a wider variety of platforms.

The idea of developing a common application interface for source code portability was not new. As the traditional UNIX system programming interface changed and diverged with each new architecture that supported the original UNIX operating system, a number of efforts arose to try and define a single operating system programming interface.

In North America, /usr/group (now UniForum), developed what was known as the /usr/group Standard in 1984. This work became input to the Institute of Electrical and Electronics Engineers (IEEE) POSIX standards committee when they formed in 1985. The original IEEE POSIX work culminated in the ratification of the POSIX.1 system interface standard in 1988 (IEEE Std 1003.1-1988), which was reformatted and became the international ISO standard in 1990. (IEEE Std 1003.1-1990 is identical to ISO/IEC 9945-1:1990, with the small exception of the IEEE book's more colorful cover.) POSIX.1 is very focused in scope, and the consensus process has created areas where the specification is either silent, or allows a choice between two different behaviors. While POSIX.1 was being developed by the IEEE, a CBEMA X3 committee developed the ANSI C standard, ratified in 1989.

As the POSIX and ANSI C efforts were developing in North America, a group of European systems vendors took a different approach to applications source code portability specifications. X/Open Company Limited was formed in 1984, as a not-for-profit organization that would be responsible for developing and maintaining specifications for applications portability that the founding share-holders would implement. X/Open produced the X/Open Portability Guide (XPG), which has been expanded and grown into an entire application environment specification known as the X/Open Common Application Environment (CAE).

As the X/Open CAE evolved, X/Open involved themselves with the IEEE POSIX process, recognizing the importance of the formal consensus process. When XPG4 was published in 1992, it was fully aligned with the POSIX.1 system interface standard and the then newly ratified POSIX.2 shell and utilities standard. (XPG4 is actually a name rather than an acronym. XPG3 did indeed stand for X/Open Portability Guide, Issue 3. By 1992, the XPG3 had evolved into the X/Open CAE, Issue 4, a much broader set of specifications. XPG4 is the convenient name for the systems interfaces and commands and utilities specifications that are part of issue 4 of the CAE.) XPG4 was a proper superset of these POSIX standards, plus the ANSI/ISO C language standard libraries, and contained a wealth of other application interfaces and utilities, many of them derived from the System V Interface Definition.

The primary sponsors of the COSE initiative were responsible for developing the common API specification. They began by defining a base for the specification using other accepted industry specifications. XPG4 was chosen as the initial specification. It was already widely implemented and used, and supported key formal portability standards in the eyes of most potential customers of this work. Next were added the traditional UNIX System V Interface Definition, (SVID) Edition 3, Level 1 Base and Kernel Extension calls, and the OSF Application Environment Specification (AES) Full Use interface definitions. These interfaces represented the stable central core of the latter two specifications.

A novel survey technique was then used for developing the common API specification. A set of key successful applications were chosen for the survey, and at either the source code or binary level these applications were inspected for system interface and library usage. The applications were chosen based on such factors as market success and application type in an effort to ensure that no particular group of applications (e.g. databases) was over-represented. A further survey of 3500 program modules was performed, recognizing that while the primary group of applications was representative of the industry, it was certainly not a complete representation. After the survey, another 130 interfaces were added to the base common API specification. Many of these were found in applications originally developed on Berkeley-based systems.

There were 1170 interfaces in the complete specification when the work was done (926 programming interfaces, 70 headers, 174 commands and utilities), and the common API specification became known as Spec 1170. The breadth and mixed origins of Spec 1170 meant there were areas of duplicated functionality. (For example, there are similar interfaces for memory management `bcopy()` versus `memmove()` and creating temporary filenames `tmpnam()`versus `mktemp()`). Such duplication was left in place to ensure that the maximum number of existing applications would be portable on systems implementing the specification. It was still the intent of the specification to provide a base for porting existing applications developed on traditional UNIX and UNIX-like operating systems. While allowing the duplication, pointers to the recommended practice for future portability were outlined. In similar fashion, the base specifications that were used were not pruned of interfaces not discovered by the survey, as it is believed they provide a clear direction for new applications development.

The specification was turned over to X/Open in December 1993 to be turned into a proper industry specification using their fast-track review process. The Single UNIX Specification is the fruit of this labor, and was published in October 1994, and the supporting X/Open UNIX brand program is in place as of March 1995. Spec 1170 is no more, replaced by the publication of the Single UNIX Specification. (There are also no longer 1170 interfaces in the specification as the fast-track review process shaped the document, and the new internationalized curses specification contributed a large number of interfaces.)

The Single UNIX Specification

The Single UNIX Specification is published as a proper superset of XPG4, and thus still has the POSIX.1 system interface, POSIX.2 shell and utilities, and ISO/ANSI C language specifications at its core. It is a collection of documents that are all part of the X/Open CAE (Common Application Environment), and consists of:

- System Interface Definitions, Issue 4, **Version 2**
- System Interfaces and Headers, Issue 4, **Version 2**
- Commands and Utilities, Issue 4, **Version 2**
- Networking Services, Issue 4
- X/Open Curses, Issue 4

The first three documents made up XPG4, and with the addition of the traditional UNIX interfaces and definitions, were re-released as Issue 4, Version 2. The only real changes to the Commands and Utilities documents were enhanced descriptions of the cc and c89 compiler front-ends. Berkeley based sockets and IP address resolution interfaces were added to the Networking Services document that already described the X/Open Transport Interface (XTI). X/Open Curses, Issue 4, is a substantially enhanced version of the Issue 3 Curses calls, supporting color, multi-byte characters, and different writing directions.

Components, Profiles, and X/Open Brands

The X/Open CAE specifications describe interfaces that developers may use when writing portable applications code. The other half of X/Open's formula for applications portability is their brand program. X/Open has identified functional components, based upon the interfaces and protocols described in the X/Open CAE specifications. For example, there is an "XPG4 Commands and Utilities V2" component, described in the X/Open CAE Commands and Utilities, Issue 4, Version 2, specification. Likewise, the "XPG4 System Interfaces and Libraries (Extended)" component is described in the X/Open CAE System Interfaces and Headers, Issue 4, Version 2, specification. There are 26 different components identified for the X/Open CAE.

Profiles are constructed from components, and there are several XPG4-based profiles.

XPG4 Base

The XPG4 Base profile was originally published in 1992. It describes a fully functional environment for portable applications development. It requires full conformance to the POSIX.1 (System Interfaces) standard, and a transitional path for POSIX.2 (Shell and Utilities). Required components are:

- XPG4 Internationalized System Calls and Libraries
- XPG4 Commands and Utilities
- XPG4 C Language

XPG4 Base 95

The current enhanced XPG4 Base profile requires full conformance to the XPG4 Commands and Utilities component. Required components are:

- XPG4 Internationalized System Calls and Libraries
- XPG4 Commands and Utilities V2
- XPG4 C Language

XPG4 UNIX

This profile is a superset of XPG4 Base 95, and describes a platform supporting the additional functions for applications portability for programs originally developed on traditional UNIX and UNIX-derived systems. Required components are:

- XPG4 Internationalized System Calls and Libraries (Extended)
- XPG4 Commands and Utilities V2
- XPG4 C Language
- XPG4 Transport Service (XTI)
- XPG4 Sockets
- XPG4 Internationalized Terminal Interfaces

The XPG4 Internationalized System Calls and Libraries (Extended) component covers all of the base interfaces, (e.g., the POSIX.1 interfaces, the ISO C library and Multibyte Support Extension addendum), makes mandatory all feature groups (POSIX.2 C-Language Binding, Shared Memory, Enhanced Internationalization), and adds the new feature group (X/Open UNIX) covering all of the new interfaces added to this version of the X/Open CAE System Interfaces and Headers volume. The feature groups are small collections of interfaces found in the System Interfaces and Headers, Issue 4, specification, that are not mandatory for implementations of the XPG4 Base and XPG4 Base 95 profiles.

To claim conformance to a component or profile, a vendor completes a Conformance Statement Questionnaire (CSQ), which is then referred to as a Conformance Statement and maintained up-to-date on file at all times. There may be authorized test suites to be run for some components, but these are simply "Indicators of Conformance." The brand agreement is the binding commitment to conformance, not the test suite. (Test suites exist for all XPG4 UNIX profile components except the new version of XPG4 Curses.)

If at anytime anybody discovers that the branded product is not behaving according to the appropriate X/Open CAE specification, the vendor is obligated to correct the problem within very strict time frames or lose the brand on their product. This has already been proven with the largest of vendors. No one committing loudly to open systems in the marketplace would want to be publicly announced as in breach of their brand contract.

There have been occasions when it would be impossible, despite the existence of a CAE specification and branding program, for any significant number of vendors to deliver products conforming to the complete specification in any reasonable time frame, and a transitional, or "soft" conformance path was allowed. This is the primary difference between the XPG4 Base and XPG4 Base 95 profiles, with the latter requiring proper conformance to the XPG4 Commands and Utilities component.

Soft branding allows the vendor to identify the specification to which they are conforming, clearly describing in a Conformance Statement Questionnaire what it is they conform to, and identifying where the vendor does not yet meet the specification, and then conform to their *statement.*

As an example, Commands and Utilities, Issue 4, was published in 1992 as part of XPG4, with the newly ratified IEEE POSIX.2 Shell and Utilities standard (IEEE Std 1003.2-1992) at its core. The specification was just different enough from every vendors' existing shell and utilities implementations that no one could really claim to conform to the XPG4 Commands and Utilities component completely and exactly as it was specified. The transitional "soft" brand was allowed for this component in this situation. Hard branding was required for the other components in the XPG4 Base profile.

Some jokingly claimed that soft branding allowed the creation of the XPG4 Commands and Utilities branded toaster, whereby a vendor could claim in their CSQ that they have a toaster that meets exactly none of the specification. While a fun example of a pathological interpretation of soft branding, obviously no vendor survives long in the marketplace with such a strategy. It does, however, point out the importance of understanding what brands are all about if you are buying X/Open branded products. Ask to see the product conformance statements, if you are comparison shopping. Soft branding is sometimes referred to as "historical" branding. Within three months of the approval of an X/Open authorized test suite, no additional requests to soft brand are accepted.

In order to clearly identify to purchasers the exact specification to which a branded product conforms, an additional attribution, called a trademark attribution, is required when reference is made to the X/Open brand and branded products or to the trademark and the product registration. The XPG4 profile attributions are:

Profile Name	**Attribution**
XPG4 Base	Base
XPG4 Base 95	Base 95
XPG4 UNIX	UNIX 95
UNIX 93	UNIX 93

So depending upon the profile to which a system was branding, they would claim: "MyProduct, Version M.m is an X/Open <Attribution> branded product"

If the system branded to the X/Open UNIX definition, described by the XPG4 UNIX Profile, (whose components are documented in the Single UNIX Specification), they would complete a Conformance Statement and deliver the appropriate test reports, and would claim: "MyProduct, Version M.m is an X/Open UNIX 95 branded product." The one "surprise" here is UNIX 93, which is an interim brandable profile. To brand this way, a system must (i) be XPG3 or XPG4 Base branded, (ii) based on USL/Novell code, (iii) pass the System V Verification Suite (SVVS), and (iv) commit to full XPG4 UNIX branding as quickly as possible. This brand is renewable for existing product versions, but no new applications for this interim brand will be accepted by X/Open 12 months after the existence of the full XPG4 UNIX brand program, i.e. beginning March 1995. The branded system may not use "UNIX" as part of the product name only in the X/Open trademark claim. Products are allowed to use "UNIX" as part of the product name if the vendor has committed in writing to obtaining full XPG4 UNIX branding within 24 months, at the time they sign the license agreement.

This "UNIX 93" interim profile is not a proper profile, and is only described in the X/Open Trade Mark License Agreement. It is only considered to be an interim measure similar to soft branding, allowing the marketplace to catch up to the specification program.

There were additional pre-existing conditions on systems registering to use the UNIX trademark, if they are derived from UNIX System Laboratories (USL) releases that constitute derivatives of UNIX System V Release 3.2, and are adapted for Intel 80x86 upwards compatible architectures. Systems that met the above criteria needed to demonstrate their ability to run a particular set of binary applications. These conditions were already attached to the UNIX trademark prior to X/Open becoming exclusive licensor, but X/Open has worked to remove these conditions.

Use of the Single UNIX Specification and X/Open UNIX Systems

With the publication of the Single UNIX Specification and the roll-out of the X/Open UNIX branding program, "UNIX" as a product (rather than a philosophy) has fundamentally changed. There is now a clear separation between the specification and trademark licensing organization, and the commercial organizations providing implementations. This is a good thing.

Novell has made the UNIX trademark available to X/Open as the exclusive licensor. The only way a system may now call itself a "UNIX" system is through the X/Open branding program. This is tied to a well-defined specification.

The Single UNIX Specification was created to ease the porting burden for applications originally developed on traditional UNIX and UNIX-like systems. X/Open's care at integrating it into the XPG4 specifications as a proper superset of XPG4 is a fundamentally good thing. XPG4 itself is a coherent well-designed specification for applications portability, based on key industry formal standards, and providing a large number of functional interfaces. It is widely implemented and used. Adding the additional calls covered by the X/Open UNIX feature group does not harm the XPG4 base portability model, and arguably enhances it.

Where functional duplication exists within the specifications, the recommended way of writing things has been clearly indicated. This is typically couched in terms of extending portability to, or remaining portable "to systems that implement previous versions of the specification," i.e., XPG4 Base functionality.

As always, application development organizations are required to understand their own needs, and design their portability model accordingly. While this is probably obvious to most readers of *;login:*, some of the sponsors' marketing teams would have you believe that the Single UNIX Specification is the greatest software development "tool" since. . . well, since sliced bread. If an organization cares predominantly about traditional UNIX environments, then shifting their application portability model to the Single UNIX Specification is likely a good idea. It means they have a well-defined, supported and maintained specification against which to write applications, and an X/Open branding program to support their systems procurements. If an organization cares about a broader portability base, including systems like DEC's VMS, then they may wish to stick with XPG4. While it may be more limited in scope, it is perhaps implemented on a broader base of systems. (Don't laugh, but IBM intends to obtain the XPG4 Base 95 brand for MVS by this September. Having worked a little with the POSIX environment provided on MVS, their intentions are good, and very realizable.)

As always, development organizations will need to ensure that the programmers understand the portability model, how to use it, and when and how to break the rules. Developers should have copies of the relevant specifications. If an organization is developing applications with the intent of distributing them on multiple platforms, they will need to understand how to organize their software management and construction environments.

The Single UNIX Specification, as with XPG4 and POSIX is simply the basis for a portability model. Development organizations still need to manage their software development environments. Those responsible for purchasing or recommending system purchases will need to understand the brand program, and their ability to request copies of Conformance Statements so they can compare potential platforms. This becomes essential if an organization is willing to accept either XPG4 UNIX 95 or UNIX 93 systems for applications development and distribution. At UniForum '95 in Dallas, TX, ten of the large traditional UNIX workstation vendors announced their UNIX 93 brands. No one has yet branded to the full XPG4 UNIX 95 profile. It will be interesting to see who will be first.

Lastly, should you need to obtain a copy of the Single UNIX Specification, there are a couple of sources. You can get a large published paper edition from X/Open directly for about $250 (contact 703 876-0044). It has also been published by Prentice-Hall in CD-ROM format with several UNIX and MS-Windows browsers and is accompanied by an introductory book, *Go Solo* (ISBN 0-13-439381-3, $70).

STANDARDS

An Update on Standards Relevant to USENIX Members

by Nick Stoughton
USENIX Standards Report Editor
<nick@hoskyns.co.uk>

The following reports are published in this column:

- POSIX.1: System API
- Language Futures
- POSIX Conformance Testing

Our Standards Report Editor, **Nick Stoughton**, welcomes dialogue between this column and you, the readers. Please send any comments you might have to *<nick@usenix.org>*.

Snitch Reports

Report on POSIX.1: System API

Nick Stoughton <nick@hoskyns.co.uk> reports on the January 16-20, 1995 meeting in Ft. Lauderdale, FL:

If You Can't Beat 'Em, Join 'Em

Well, after tiring of twisting people's arms (usually unsuccessfully) to snitch on the work of the POSIX.1 Working Group, where I am usually to be found during a POSIX meeting myself, I thought I would have to submit at least one real snitch report myself. Show everyone how its really done. . . but the only trouble is, I have all those other great snitches on the other groups to live up to!

POSIX.1 is a small group; typically only about 6 or 7 show up at a meeting. Until this meeting, it had only two items of work: POSIX.1a and Interpretation requests. However, a new Project Authorization Request for removable media has just been passed and assigned to this group, so perhaps I'll have some more people to dragoon in April!

Another Year, Another Draft

The great highlight of the January meeting was that Lee Damico, our technical editor, had managed to ship draft 12 of POSIX.1a to the IEEE a week before the meeting. This was after many late nights, weekends, and even sacrificed Christmas vacation trying to get all the ballot resolutions into the draft.

Several sections were substantially rewritten after the first round of balloting; most notably the checkpoint-restart interfaces. These are highly contentious for a number of reasons:

- They are not based on existing practice, and are substantially invention by the committee. As a USENIX and EurOpen (the European equivalent of USENIX) representative, this makes me decidedly uneasy. However, they are optional, and I don't believe they are unimplementable. Let us hope some brave reader tries to build these interfaces before the standard is published so we can have some existing practice and experience to work with!

- Two different groups within POSIX require them for future extensions. The Batch profile, which was responsible for wording the original (up to Draft 11) version, needs some interfaces to allow long running jobs to be stopped in their tracks and restarted later. In addition, the work being done by the Services for Reliable, Available, Serviceable Systems (SRASS) working group needs interfaces to checkpoint processes if the system is failing. This would allow those processes to be restarted later in, potentially, quite a different environment after various devices have disappeared.

The batch and SRASS arguments look set to roll on for some time; the two groups want quite different things. However, it is also undesirable to have two different checkpoint interfaces: one for batch, one for SRASS. The ballot group just hates the whole idea because it is invention. But if we remove the interfaces, then these two follow-on groups are in trouble.

The other recurrent long-running argument in POSIX.1 is over the interpretation of trailing slash characters on pathnames. Historically, System V based systems ignored all trailing slashes, whatever. BSD based systems only permitted trailing slashes on pathnames that were existing directories. In draft 12, the BSD behavior is demanded. From an engineering perspective, this is a much cleaner answer. However, the weight of existing System V platforms that would lose compliance as a result of this is substantial.

From the application developer's, as opposed to the system implementor's, point of view, it is a good change. If you wrote an application that depended on trailing slash behavior before POSIX.1a, it was not portable. After the publication (well, we have to be optimistic), such an application *will* be portable.

For most system implementors it is not too hard a change to get working in a future release, and most have at least one release a year.

News Flash

As ballot coordinator for POSIX.1, I can tell you that the ballot on draft 12 closed on time, with a 44% affirmative vote. Most of the ballots look as if we should be able to work through them quickly, so progress can be expected in April.

Report on Language Futures

John L. Hill <HILL@po3.bb.unisys.com> writes on Language Futures

Abstract

This article declares that there are at least two things that the industry could, indeed should, pursue vigorously. Those are language independent application programming interfaces and formal description techniques. The information technology (IT) industry needs to develop them further and employ them more broadly in order to maintain the current rate of improvement in software technology.

The Setting – Abstraction is the Root

From a historical perspective, the art of software development is progressing to become a mature science. The progression (from art to science) is becoming increasingly visible because there is only now beginning to emerge the precepts of a science. First, the IEEE began a movement to establish software engineering as a formal, certifiable profession. The existence of a formal profession is one mark distinguishing art from science. Software engineering is beginning to reveal its maturity. Second, a body of standard, formal definitions, and agreed upon principles is emerging. As time passes software engineering should continue to lose much of its mystique. It will grow in strength as monumental changes take place.

The original "hard-wired" but variable program computers gave way to truly soft programmable computers. At first the languages in which one wrote programs were "close to the machine." Early languages, notably machine code and assembly languages, required a programmer to know the internal structure and operation of the specific target machine. Limitations and difficulties revealed themselves in droves. Assemblers were not applicable to any but the single, target computer architecture. This limitation of portability affected not only the program but the programmer as well. Yet another complication was that the programs were often so "personalized" to both the target machine and the programmer, and difficult to understand, as to not be maintainable by anyone but the author.

With the inexorable passage of time, champions of software engineering dealt with the weaknesses of those early generation programming tools. They invented higher order languages. The high order languages masked programmers from the internal workings and structure of the computers on which they were working. It was simply the application to computers of something the world's great painters and sculptors embraced many decades before. . . abstraction. By way of analogy, I assert Grace Hopper was the logical equivalent of Mondrian in terms of abstraction. In concentrating on the essence of what the application required, rather than the means of satisfying those requirements, the programmer was relieved of "unnecessary" computer operational details.

In the visual arts an entity's form, color, and texture are the abstractions of its physical representation. Programming languages are similar in terms of abstraction. Executing a single verb often invokes multiple primitive operations in the computer. By making those verbs natural-language-like, the programs themselves became easier for mere humans to write, reread, and maintain. Data manipulation became simpler, too, with the appearance of more and varied data types coupled with the verbs to manipulate them reliably.

Another aspect of abstraction is becoming evident in computer operating systems. Operating systems are becoming increasingly modular. Functionally different elements are separating from the core operation system and from one another. As a by-product, this provides the means for functionally expanding and tailoring individual OS elements. The current evolution of middleware is a representative example.

Yet another leap is taking place in software engineering technology with the truly generative languages. These are frequently referred to as fourth generation languages (4GLs). 4GLs allow the programmer to functionally describe the application to be programmed and leave the generation of the traditional high order language program to the 4GL.

As these improvements in software programming tools and languages are taking place, the technologies that are used to implement them changed, too. Improvements have come rapidly to the physical computer system itself. Many of these important advancements have not greatly affected the programmer since programmer activities tend to take place at a higher level of abstraction. The result is that the programmer largely remains insulated from the rapid advancements taking place in computer hardware. There are notable exceptions, such as self-identifying peripherals and dynamic configuration. These allow the programmer to be virtually ignorant of the physical devices and their physical, and often their logical relationships. Some changes in computers that challenged programmers include new input and output devices as well as communications protocols and media.

Programmers, resilient by nature, have readily adapted to all these improvements. There is a large challenge, however. That challenge is in retaining the current rate of improvement in software engineering within a framework of continually changing hardware and related platform capabilities. As has been the case previously, the solution lies in abstraction.

There are two areas in which the application of abstraction will benefit the IT industry. They are language independent methods of specifying application programming interfaces, and formal description techniques. Adoption of the principles of these will yield enormous dividends.

Language Independent Application Programming Interfaces

An application programming interface, or API, is that artificial boundary over which an application program obtains a service it requires in order to obtain some result. A representative example can be found in the programming language binding to a database manager. The Ada binding to SQL is one such API. By a programmer using that API, the Ada programmer has access to the extensive data management services of SQL. Without that access, the programmer would have to perform data management functions within the program itself.

As powerful and useful as the Ada/SQL binding is, it and others like it, suffer from language dependency. That is, the binding is good only for Ada. One likely consequence is the sacrifice of interoperability in a mixed language environment. Perhaps a bit of explanation is desirable. If the environment of the SQL database is diverse, as far as the programming languages in use to access its data are concerned, then the syntactic capabilities of those languages' bindings will differ. Because programming languages very frequently differ semantically, the functionality of the bindings will differ. This results in reduced interoperability when viewed from the point of view of the SQL database.

The application of the concept of abstraction leads to one way of avoiding the sacrifice. If, instead of developing language specific APIs, one develops a language independent specification of the API and then develops language specific bindings to that language independent specification, then the bound service (in the instance of the example SQL is the service) could reliably exist in a mixed language environment. This preserves interoperability.

Naturally, there are difficulties with the implementation of this proposal. First is the methodology for expressing the language independent specification. The methodology must be sufficiently robust to encompass the broadest range of programming language paradigms. The IEEE's standards development committee for POSIX developed one effective methodology. Its application outside the POSIX environment is certainly possible.

Second is the actual development of the language independent specifications themselves. Because there is no direct, financial value to having a particular language-independent specification (after all, the language dependent binding is valuable to the programmer and not the language independent specification to which it is bound) it is difficult to justify development. The IT industry needs an economically viable means for developing language independent specifications.

Formal Description Techniques

A formal description technique, FDT, is a mathematically precise (complete with proofs) means of semantically describing systems. Systems described using FDTs exhibit properties such as symmetry and completeness. The use of FDTs to describe computer programming languages and applications will result in high-quality, low-defect implementations.

As was the case with language independent APIs, there are some practical difficulties with FDTs. First is the syntax used by the FDT. There are currently several projects, at various stages of completion, developing standards for FDTs. Examples include ASN.1, LOTOS, and ESTELLE, each targeting a portion of the data communications realm. Both VDM-SL and Z, while earlier in the development process than their communications counterparts, apply to the programming languages and diverse applications realms. Second is the lack of product backing so necessary to broad-based development. There are few product possibilities stemming from FDTs and they are highly specialized. The value of FDTs derives not from the tools that it uses but in the quality of the applications it produces.

Conclusion

The IT industry today is developing two very valuable tools, language independent APIs and FDTs, albeit at a low priority. Their value is indirect. They are complicated and highly specialized. Their value, though, is undoubtedly high having extensive possibilities for implementation. In order for the world to continue the rate of improvement in software engineering technology, the IT industry needs to heartily embrace them.

Report on POSIX Conformance Testing

Kathy Liburdy <kliburdy@eng.clemson.edu> reports on the January 16-20, 1995 meeting in Ft. Lauderdale, FL:

Should Test Methods Be Standardized?

A point of contention which arose during the rebellion against required test methods reappeared in the January POSIX meeting: Should test methods be standardized at all? Although working groups are no longer required to develop test methods, the proper dress for test methods has yet to be established.

In the midst of concerns focused on testing (e.g., 2003.5 conformance to 2003), members of the POSIX.5 Ada working group reconsidered the role of testing in the Big Picture of open systems standards. While the POSIX.5 working group unanimously agreed that a testing capability was needed for the POSIX Ada Language Interfaces, there were concerns about the standardization of test methods. The following points were raised by members of the group:

1. Standardization of the test methods invites problems with interpretations in the event the base standard disagrees with the test method standard.
2. Standardization of the test methods prior to actual use violates the conventional wisdom of standardizing based on "existing practice."
3. The benefits of test method standardization are unclear, while the disadvantages (e.g., the time and labor involved) are painfully obvious.

A meeting with representatives of the POSIX testing community was requested in hopes of resolving, or at least understanding, these issues. Both Roger Martin, chair of 2003 and the Sub Committee on Conformance Testing (SCCT) and Lowell Johnson, chair of the IEEE Portable Applications Standards Committee (PASC), attended this meeting. During the discussion which circled these issues, the idea of publishing 2003.5 as a Recommended Practice was proposed as a sort of multi-purpose compromise which would:

- keep the test methods in the official PASC process,
- remove any doubt about "who wins" in a disagreement with the base standard,
- remove concern about conformance with 2003 (since the scope of 2003 is test methods "standards"), and
- serve as an experiment to help determine the appropriate role of test methods in IEEE POSIX.

This suggestion was met with general favor, and the POSIX.5 working group agreed to submit a request to the Sponsor Executive Committee (SEC) to modify the status of 2003.5 from that of a Standard to a Recommended Practice.

What are Test Methods?

In order for the POSIX community to reach consensus on the appropriate niche for test methods (e.g., standard, recommended practice, guide, or none of the above), attention must be given to uncloaking the meaning of this much used and ill defined term.

According to the January draft of 2003, test methods are "software, procedures, or other means to measure conformance. Test methods may include a PCTS, PCTP or an audit of PCD." (For the nonconformance tester, PCTS is the POSIX Conformance Test Suite, PCTP is the POSIX Conformance Test Plan, and PCD is the POSIX Conformance Document). In practice, the use of test methods may refer to the actual implementation of test methods or the specification of test methods. While the stated scope of 2003r is test methods, the emphasis is almost entirely on the development of assertions about required behavior of a conforming implementation. However, the definition of test methods makes no explicit mention of assertions.

Ballot objections to 2003r reflect this confusion and concern: what exactly are test methods? Many of the ballot objections submitted against 2003r address fundamental definitions such as test methods and assertions, and question the intuitiveness of recently crafted definitions. As an example, an assertion is defined to be a test specification. The terminology "assertion test" is then introduced to mean the executable form of the assertion, requiring the reader of the document to comprehend a "test specification test." The wobbly state of the terminology reflects the immaturity of the area of conformance testing, which should send a bright, loud, blinking warning to anyone considering the merits of standardizing test methods.

Study Group Formation for Automated Testing

The initial concern in the development of automated test methods for the Ada Language Interfaces was for achieving conformance to 2003r. This concern was well-founded, since 2003r embraces a manual approach to test development. As the interest in automated testing increased, the question of conformance to 2003 began to be replaced by the question: "Is there an approach for developing test specifications based on this experience which could be useful to others embarking on a similar effort?"

This question, with an affirmative answer, was brought before the SCCT/2003 group for discussion. Two possible alternatives were identified:

- include an automated testing methodology in 2003r, or
- submit a PAR for a separate document to address automated testing

Since 2003r ballot resolution had not been completed, and the issues related to the requirements of an automated testing methodology would benefit from an open review, the group approved the idea of requesting a study group for the April meeting. The purpose of the study group is to determine the requirements of automated testing, and to evaluate 2003r with respect to these requirements. The request for an Automated Testing Study Group was brought before the SEC and approved.

Documents to be considered by the study group include 2003r, the current draft of 2003.5, the Clemson Automated Testing System Language Reference Manual and the Assertion Definition Language (ADL) Reference Manual. Members of Sun Microsystems Laboratories are scheduled to present recent updates and experiences with ADL. Because of the relationship between automated testing and formal specification languages, a representative of the X3J21 Committee on Formal Description Techniques is scheduled to provide an overview of ongoing work in formal methods. Suggestions for additional topics are invited.

Dilbert ® by Scott Adams

THIS VIDEO PHONE WILL BE A BIG HELP FOR DATING.
S.Adams
THIS WAY I CAN WEED OUT THE UNATTRACTIVE PROSPECTS IN SECONDS.
6-2
© 1992 United Feature Syndicate, Inc.
ISN'T THERE A CAMERA ON YOUR END TOO?
NO SYSTEM IS PERFECT.

DILBERT reprinted by permission of UFS, Inc.

The Bookworm

by Peter H. Salus
<peter@uunet.uu.net>

Satire is something that warms the heart. I was at FOSE in Washington where a well-suited young man told me that his product, which I will leave nameless, was "open systems," "scalable," and "POSIX-compliant." I was tempted to ask him what that was supposed to mean. Instead, I asked whether that meant all parts of POSIX, as I was particularly interested in .5 [for those of you who don't care, that's the Ada bindings]. "Oh, yes," twinkle-toes informed me. "All *de jure* and *de facto* standards." I thanked him and forged on to look for free tee-shirts. He needed his definitions fixed. Like:

Pentium: A hot chip in every way, melting your mother's heart and board.

APL: A language so compacted that the source code can be freely disseminated without revealing the programmer's intentions or jeopardizing proprietary rights.

In 1981 I thought that Stan Kelly-Bootle's *Devil's DP Dictionary* was the best thing to read after works by Thurber, Perelman, or Woody Allen. After a long spell in which I could only get measured amounts of SK-B in *UNIX Review*, MIT Press has come out with a successor, *The Computer Contradictionary.* It's much more than a mere updating. The presence of the Pentium quip (above) shows that it's fairly *au courant*, though not enough so to feature an SK-Bian comment on Windows NT.

This is a wonderful piece of satire with over 1000 mis-definitions. But it's also real history: The comments on APL, FORTRAN, C, C++, Edward M. Cherlin, and Luddite (among others) yield info as well as smiles.

My favorites? **DOS**: a series of fatal PC viruses distributed by Microsoft; and **POSIX:** Presumed recursive acronym for POSIX Is not uniX.

But I envy SK-B. I think of myself as a curmudgeon, but he certainly out-curmudgeons me.

Buy this book.

ORA's stars

I was going to review Don Libes' *Exploring Expect*, but Glen Vandenburg beat me to it. Don gave a paper on *expect* at the Anaheim (1990) USENIX, Mike O'Dell and I persuaded him to write it up and published his article in *Computing Systems* 4.2. It's a great tool and this is a fine book.

Linux is an admirable system. My concrete (non-subjective) evidence for this is that someone stole my installation and getting started book (reviewed here last October). Well, I guess it doesn't matter for here's a volume on *Running Linux* by Matt Welsh and Lar Kaufman.

This is a solid book which features hardware and systems administration as well as programming and networking information. It's written in a matter-of-fact manner and is chock-full of information. It's the perfect complement to ORA's *Linux Network Administrator's Guide.*

It may be that POSIX Isn't uniX, but P1003 has some concrete products: .0 (the Guide); .1 (API); .2 (Shell and Tools); .5 (Ada binding); and .4 (Real Time). Bill

Books reviewed in this column:

Stan Kelly-Bootle, **The Computer Contradictionary**. MIT Press, 1995. ISBN 0-262-61112-0. $14.95.

Don Libes, **Exploring Expect**. O'Reilly & Associates, 1995. ISBN 1-56592-090-2. $29.95.

Matt Welsh and Lar Kaufman, **Running Linux**. O'Reilly & Associates, 1995. ISBN 1-56592-100-3. $24.95.

Olaf Kirch, **Linux Network Administrator's Guide**. O'Reilly & Associates, 1994. ISBN 1-56592-087-2. $24.95.

Bill O. Gallmeister, **POSIX.4: Programming for the Real World**. O'Reilly & Associates, 1995. ISBN 1-56592-074-0. $29.95.

Linda Branagan and Mike Sierra, **The Frame Handbook**. O'Reilly & Associates, 1994. ISBN 1-56592-009-0. $24.95.

Balachander Krishnamurthy, ed., **Practical Reusable UNIX Software**. John Wiley & Sons, 1995. ISBN 0-471-05807-6. $34.95.

Stephen C. Dewhurst and Kathy T. Stark, **Programming in C++**, second ed. Prentice Hall, 1995. ISBN 0-13-182718-9. $28.00.

Jack Hodges, **An Introduction to Berkeley UNIX and ANSI C**. Prentice Hall, 1995. ISBN 0-13-068495-3.

Amir Afzal, **UNIX Unbounded**. Prentice Hall, 1995. ISBN 0-13-328246-5.

Gallmeister, formerly of Lynx, and vice-chair of 1003.4, has come up with a reliable and useful book on real-time computing (which he calls "Computing for the Real World"). It should serve as a deskside reference for anyone involved with real-time applications.

The Frame [4.0] Handbook by Branagan and Sierra appears to be another useful and accurate handbook. I am not a Frame user, so I didn't test it out. But it reads well and makes sense.

I'm not sure what ORA has in the works, but I've been using `zsh` for the last few months and find the on-line manual lacking. As there are several `csh` and `ksh` books, I'd think it was time for one on `zsh` and another on `bash`.

I have a minor complaint where ORA is concerned. After moving from their sturdy brown covers to the animal kingdom, I'm genuinely saddened that they've now abandoned the zoosphere. I was hoping for (say) a POSIX.1 book with a unicorn or wyvern or just an unau.

Reusable Software

In 1978 we had the "blue" issue of the *BSTJ*, in 1982, there was the "yellow" volume. Now Wiley has brought out an anthology of papers that reads as though it were a 1995 issue of that journal. Edited by Krishnamurthy, with a brief foreword by Dennis Ritchie, this volume's authors include such unknowns as David Korn, John Snyder, David Rosenblum, Steve Bellovin, Kiem-Phong Vo, etc., etc. The articles are about "Libraries and File System Architecture," "Configuration Management," "Tool- and Application-Building Languages," "Reverse Engineering," "Software Security," "Fault Tolerance," "Generalized Event-Action Handling," "Intertool Connections," and "Monitoring." There are overviews at both the beginning and the end.

Though it is not about a specific program or set of programs, I found Bellovin's short essay on security superb. In just over a dozen pages, Bellovin encapsulates a great deal of out-and-out wisdom.

Bell Labs has been turning out useful software for nearly three decades. I especially liked the descriptions of ast, nmake, Easel, Yeast, and Marvel. Some of the work has been spoken about at USENIX conferences and workshops. It's great to have it all together in one place.

C++

There is a new edition of Dewhurst and Stark's *Programming in C++*. It is nearly 100 pages bigger than the 1989 version. Solid and useful, it is highly recommended.

Raspberries

The second sentence of Hodges' textbook, *An Introduction to Berkeley UNIX and ANSI C*, reads: "This book is intended to introduce you to a widely used version of UNIX called Berkeley System Distribution, or Berkeley UNIX" (xi). Afzal's book, *UNIX Unbounded*, (p. 24) informs me that "This version of UNIX is called the Berkeley Standard Distribution (BSD)."

Sigh.

Look at the title pages of the manuals, folks. The 4.2, 4.3 and 4.4 sets all say **Berkeley Software Distribution**. Can it be that the folks who write these textbooks never looked at the manuals? Do the authors not even know what they're writing about? I barely read the rest of either volume. Why would I bother?

What confidence could I have in what they tell me?

I showed the two passages to Mike O'Dell. He advised me to throw out the books. This to Peter the Curmudgeon.

The Early History of Data Networks

By Gerard J. Holzmann and Bjorn Pehrson (IEEE Computer Society Press, 1995, ISBN: 0-8186-6782-6.) 291pp.

Reviewed by George V. Neville-Neil <gnn@netcom.com>

With the current craze for the Information Superhighway, some might think that the idea of building data networks is a 20th century phenomenon. Those who remember their history lessons might point as far back as the 19th century to the telegraph or the pony express.

The truth of the matter is that data networks have existed for even longer. Since the earliest days of civilization people have realized how important it is to transmit news and information quickly. This book exposes the efforts of people from the times of the Greeks and Romans through the 17th and 18th century to build communication networks.

This book, written in an easy-to-read style, is a fascinating discussion of signaling methods from crude fires to the height of 18th Century engineering, the semaphore. The second and third chapters are dedicated to the two most notable data networks of the time. One, built in France by Claude Chappe and his brothers, is discussed in Chapter 2. The other, built by Abraham N. Edelcrantz in Sweden, makes up Chapter 3. Both of these systems were Optical Telegraphs. That is, they were set up so that a viewer at one station could see the signals at the next. These networks were actually quite extensive for a period – especially the French network, which at one time reached 27 cities from Amsterdam in the North, to Narbonne in the South, Brest in the West and Venice in the East.

Chapter 4 is a translation of Abraham Niclas Edelcrantz's "A Treatise on Telegraphs." It completely describes his work on the Swedish network, including coding tables, diagrams of the hardware used to construct the telegraphs, and comparisons with other designs of the time.

Other countries are covered in Chapter 5. Though no other country seems to have made the effort that the French and the Swedish did, there were designs and implementations of optical telegraph networks in many countries throughout Europe, and in North America. Though optical telegraphy was not widely used in the United States, the famous Telegraph Hill in San Francisco is named after the fact that it served as one of three network stations that were used between 1849 to 1853 to announce the arrival of ships.

Chapter 6 wraps up the book with a discussion of the nature of invention. One of the themes that runs through the book is that although many people come up with an idea, few have the gumption to get it done. The thesis here seems to be that invention is really ninety-nine percent perspiration.

After reading all of this you might be asking, "But why should I buy this book? It seems more of a curiosity than something useful." I think that there are many reasons to buy and read this book. The first, of course, is that it is an enjoyable and interesting read. I found that it really caught my interest, because these people were trying to solve the same basic problems I deal with every day in a completely different era, and with completely different tools.

For instance, then as now security was a great concern. An account is given of some bankers who use data spoofing (sending incorrect messages about the state of the stock market in Paris) to make money. The tools they use to do this aren't technical, they just bribe one of the people working for the network to transmit false messages.

In my opinion, *The Early History of Data Networks* is an excellent read, and an excellent addition to any network programmer's shelf.

Casting the Net: From ARPANET to Internet and Beyond

by Peter H. Salus (Addison-Wesley, 1995, ISBN 0-201-87674-4.) 297 pp. $26.95

Reviewed by W. Richard Stevens <rstevens@noao.edu>

Exactly one year ago I read Peter's earlier book *A Quarter Century of UNIX* and liked it a lot. Although I had been using UNIX for over half of that quarter century, I was not at Bell Labs or Berkeley, so lots of the information, personal stories, and anecdotes were new. It was a delightful read that I finished in two sittings.

Peter's new book on the history of the ARPANET and its evolution into the Internet is similar for me: although I have been using the net for many years, I was not at UCLA or BBN and was not involved in its development. This new book is also a delightful read and an obvious successor to his previous book. It is a book that needed to be written and Peter has done an excellent job, talking to the people actually involved in all the original work.

The book is divided into five parts, based on a time line. Part 1 is 1940-1964 (laying the groundwork for packet switched networks), part 2 is 1967-1972 (the start of the ARPANET), part 3 is 1974-1981 (first paper on TCP/IP, UUCP, Usenet, ISO-OSI reference model), part 4 is 1982-1989 (EUnet, switchover of the ARPANET to TCP/IP, NSFNET backbone, UUNET, ARPANET shutdown), and part 5 is 1988-1994 (NSFNET T1 to T3 backbone, archie, WAIS, gopher, WWW, EBONE, GOSIP, etc.). The coverage of all the various topics was fine for me, but I'm sure people involved in any one of the small areas might complain that their area didn't get enough pages.

My only complaint about the book is its heavy use of acronyms and its lack of an acronym glossary. There are bound to be some acronyms not familiar to the reader and it's sometimes hard to find in the surrounding text just what the acronym stands for. But that's a minor blemish on an otherwise enjoyable book. I highly recommend this book for anyone interested in just how the Internet got to be what it is today.

USENIX Conference on Object-Oriented Technologies (COOTS)

June 26-29, 1995
Monterey, California

Tutorials:

Programming Distributed Objects: The Modula-3 Approach
Geoff Wyant, Sun Microsystems Laboratories

Microsoft's OLE and COM
David Chappell, Chappell and Associates

The C++ Standard Library
Michael Vilot, ObjectCraft, Inc.

Building Distributed C++ Systems Using OMG CORBA and Object Services
Steve Vinoski, Hewlett-Packard

Design Patterns: Elements of Reusable Object-Oriented Software
John Vlissides, IBM Research

Component-based Software Development with Oberon
Wolfgang Pree, University of Linz and Josef Templ, ETH Zurich

Building Reusable Components in C++ Using OLE and COM
Don Box, DevelopMentor

Introduction to the SOMobjects Toolkit and Metaclass Programming
Ira R. Forman, IBM

Concurrent Object-oriented Network Programming with C++
Douglas C. Schmidt, Washington University, St. Louis

A Programmer's Guide to Fresco
Mark Linton, Silicon Graphics, Inc.

Technical Program:

Wednesday, June 28, 1995

Opening Remarks
Vincent F. Russo, Purdue University

Keynote Address: Object Technologies: Where are we? Where are we going? Are we lost?
Michael L. Powell, Chief Architect of Object Products, SunSoft, Inc.

Distributed Object Systems I

Session Chair: *Doug Lea*

Simple Activation for Distributed Objects
Ann Wollrath, Geoff Wyant, and Jim Waldo, Sun Microsystems Laboratories

Dynamic Insertion of Object Services
Ajay Mohindra and Murthy Devarakonda, IBM T.J. Watson Research Center; George Copeland, IBM Austin

Object-Oriented Components for High-speed Network Programming
Douglas C. Schmidt and Tim Harrison, Washington University, St. Louis

Tools to Cope with Complexity

Session Chair: *Murthy Devarakonda*

Program Explorer: A Program Visualizer for C++
Danny B. Lange and Yuichi Nakamura, IBM Research, Tokyo Research Laboratory

Configuration Management in an Object-Oriented Database
Mick Jordan and Michael Van DeVanter, Sun Microsystems Laboratories

Debugging Storage Management Problems in Garbage-Collected Environments
Dave Detlefs and Bill Kaslow, Digital Equipment Corporation, Systems Research Center

Panel Discussion: Active Content Object-Oriented Languages for the Web (Cool Languages)
Panel Chair: *Ted Goldstein*
Panelists: *To Be Announced*

Thursday, June 29, 1995

Object-oriented Languages

Session Chair: *Luca Cardelli*

Phantom: An Interpreted Language for Distributed Programming
Antony Courtney, Trinity College, Dublin, Ireland

A Framework for Higher-Order Functions in C++
Konstantin Laufer, Loyola University of Chicago

Lingua-Franca: An IDL for Structural Subtyping Distributed Object Systems
Patrick Muckelbauer and Vincent F. Russo, Purdue University

Panel Discussion: COM and CORBA
Panel Chair: *Mark Linton*
Panelists: *To Be Announced*

Distributed Object Systems II

Session Chair: *Jim Waldo*

Adding Group Communication and Fault-Tolerance to CORBA
Silvano Maffeis, Cornell University

Using Meta-Objects to Support Optimization in the Apertos Operating System
Jun-ichiro Itoh, Keio University; Yashuhiko Yokote and Rodger Lea, Sony Computer Science Laboratory, Tokyo

The Spring Object Model
Sanjay Radia, Graham Hamilton, Peter Kessler and Michael L. Powell, SunSoft, Inc.

Object Potpourri

Session Chair: *Christopher Pettus*

Integration of Concurrency Control in a Language with Subtyping and Subclassing
Carlos Baquero, Rui Oliveira and Francisco Moura, Universidade do Minho, Portugal

Generic Containers for a Distributed Object Store
Carsten Weich, Universitaet Klagenfurt, Austria

Media-Independent Interfaces in a Media-Dependent World
Ken Arnold, Sun Microsystems Laboratories; Kee Hinckley, Utopia, Inc.; Eric Sheinbrood, Wildfire Communications

To register, call the USENIX conference office at 714 588-8649.

Third Annual Tcl/Tk Workshop

July 6-8, 1995, Toronto, Canada

Thursday, July 6, 8:30-5:30

Introduction to the workshop

Tcl-DP Name Server
Peter T. Liu, Brian Smith, and Lawrence Rowe, University of California, Berkeley

Multiple Trace Composition and Its Uses
Adam Sah, University of California, Berkeley

TclProp: A Data-Propagation Formula Manager for Tcl and Tk
Joseph A. Konstan, University of Minnesota

Advances in the Pad++ Zoomable Graphics Widget
Benjamin B. Bederson and James D. Hollan, University of New Mexico

A Table-based Layout Editor
George G. Howlett

Mega-Widgets

Mega-widgets in Tcl/Tk
Shannon Jaeger, University of Calgary

Designing Mega Widgets in the Tix Library
Ioi K. Lam, University of Pennysylvania

[incr Widgets] An Object-Oriented Mega-Widget Set
Mark L. Ulferts, DSC Communications Corporation

Tcl/Tk Updates and Futures
John Ousterhout, SunLabs

Open Discussion

Cross Platform Support in Tk
Ray Johnson and Scott Stanton, Sun Microsystems Laboratories

Language Issues

Automatic Generation of Tcl Bindings for C and C++ Libraries
Wolfgang Heidrich, Computer Graphics Laboratory

An Anatomy of Guile: The Interface to Tcl/Tk
Thomas Lord, Cygnus Support

A Tcl to C Compiler
Forest R. Rouse and Wayne Christopher, ICEM CFD Engineering and the University of California, Davis

Friday, July 7, 8:30-5:30

Using Tcl/Tk to Program a Full Functional Geographic Information System
George C. Moon, Alex Lee, Stephen Lindsey

TkReplay: Record and Replay for Tk
Charles Crowley, The University of New Mexico

PLUG-IN: Using Tcl/Tk for Plan-Based User Guidance
F. Lonczewski, Munich University of Technology

A Graphical User Interface Builder for Tk
Stephen Uhler, Sun Microsystems Laboratories

Panel: Tcl and Tk in the Classroom: Lessons Learned
Joseph A. Konstan, University of Minnesota

Object-Oriented Extensions afternoon

The New [incr Tcl]: Objects, Mega-Widgets, Namespaces and More
Michael J. McLennan, AT&T Bell Laboratories

Objective-Tcl: An Object-Oriented Tcl Environment
Pedja Bogdanovich, TipTop Software

Extending Tcl for Dynamic Object-Oriented Programming
David Wetherall, MIT Laboratory for Computer Science

Interpreted C++, Object Oriented Tcl, What next?
Dean Sheehan, IXI Limited

When is an object not an object?
Mark Roseman, University of Calgary

Saturday, July 8, 8:30-5:30

Media session

Tcl Commands as Media in a Distributed Multimedia Toolkit
Joseph A. Konstan, University of Minnesota

Taming the Complexity of Distributed Multimedia Applications
Frank Stajano and Rob Walker, Olivetti Research Limited

Plug-And-Play with Wires
Max Ott, NEC USA Inc.

RIVL: A Resolution Independent Video Language
Brian C. Smith, Cornell University

Applications

Two Years with TkMan: Lessons and Innovations
Thomas A. Phelps, University of California, Berkeley

Prototyping NBC's GEnesis Broadcast Automation System Using Tcl/Tk
Brion D. Sarachan, Alexandra J. Schmidt, GE R&D Center, Steven A. Zahner

Customization and Flexibility in the exmh Mail User Interface
Brent Welch, Xerox Parc

Experience with Tcl/Tk for Scientific and Engineering Visualization
Brian W. Kernighan, AT&T Bell Laboratories

Tcl Extensions for Network Management Applications
J. Shonwalder and H. Langendorfer, Technical University of Braunschweig

Plan next year's conference

Registration Information

The workshop is open to the entire Tcl/Tk community. Registration for the workshop will cost $250. This includes a copy of the proceedings, lunches, coffee, snacks and a reception/dinner. Our experience has shown that an exchange of ideas works best when there is a small to moderate number of participants. For this reason, we are interested in bringing together active users of Tcl/Tk. Because of this and space constraints, *we will be limiting participation in this workshop to 150 people.*

If you would like to attend the workshop, please submit a short outline (no more than 1/2 a page) describing your reason for attending the workshop. This request should also include the following information:

First and Last Name
Company or Institution
Mailing address
Telephone number
Internet address
Any special dietary requirements
Any special needs (e.g. handicap)

Registration deadline: early June

Upon acceptance for attending the workshop, you will receive instructions for payment as well as information for car rental and GST rebate. Registration can be submitted electronically by sending email to: <*w95@system9.unisys.com*> or by mailing/faxing to:

Tcl/Tk Workshop 95
c/o Unisys Canada Inc
61 Middlefield Road
Scarborough, Ontario, M1S 5A9
Canada
Fax 416 297-2520

The workshop is being held at the Royal York Hotel. A special room rate has been given to the workshop of $119 Canadian, plus taxes. For this rate, reservations must be made before June 1, 1995 and you should reference the Tcl/Tk workshop. The hotel address is:

Royal York Hotel
100 Front Street
West Toronto, Ontario, M5J 1E3
Canada 800 441-1414 (inside North America)
416 368-2511 (elsewhere)

The workshop is sponsored by Unisys Inc. and the USENIX Association.

Program Committee

Co-chairs:
Ben Bederson, University of New Mexico
<*bederson@cs.unm.edu*>
Will Wilbrink, Unisys Inc.
<*will@system9.unisys.com*>

John Ousterhout, Sun Microsystems, Inc.
Steve Uhler, DSP Group
Gerald Lester, Computerized Processes Unlimited
Charley Crowley, University of New Mexico
David Richardson, University of Michigan
Wayne Christopher, ICEM CFD Engineering
Kevin Kenny, General Electric Corporate R&D
Brian Smith, Cornell University

LISA '95: The Ninth USENIX Systems Administration Conference

Be sure to mark your calendar for September 18-22 if you plan to attend LISA in Monterey, California.

The theme for this year's program is "New Challenges" which includes emerging issues such as integration of non-UNIX and proprietary systems and networking technologies, distributed information services, network voice and video teleconferencing, and managing very complex networks. There is a special emphasis in technical papers reflecting hands-on experience.

For seasoned system administrators, a special workshop, "Advanced Topics in System Administration" is being offered on Tuesday, September 19. This one day event, organized by John Schimmel of Silicon Graphics, will focus on a discussion of the latest-breaking technical issues that are raised by the attendees. Attendance is limited and based on the acceptance of a position paper. Acceptance notices will be issued by August 14.

Potential attendees may submit a proposal of at most three pages (ASCII) via email to <*jes@sgi.com*> no later than August 1. The proposal should contain a topic for discussion; a description of the subject, explaining why this topic is controversial or interesting, and a personal position. A subset of the positions will be discussed in an open forum.

Participants must be registered for the LISA conference. There is no additional fee to attend this workshop, and lunch will be provided.

Registration materials for the entire conference will be available in July.

USENIX Tutorials on Electronic Commerce

July 11-14, 1995
New York, New York

If your job involves any aspect of electronic commerce, mark your calendar with these dates: July 13-14. USENIX will offer up to six half-day tutorials on two days (some will be repeated) on electronic commerce at the Sheraton New York Hotel and Towers in New York City. While the agenda has not been finalized, topics will include:

- The Law of Electronic Commerce (EDI and E-Mail Contracts, records and privacy)
- Security/Firewalls
- Cryptography
- Survey on payment mechanisms
- Agents: active and passive, security
- Promoting your Internet business
- Creating Web documents with HTML

Three ways you can get registration information:

1. Call: 714 588 8649
2. Fax: 714 588 9706
3. Email: *conference@usenix.org*

The complete program is also available on the Web (URL: *http://www.usenix.org*).

Registration materials will be available in May.

Student Benefits and Outreach Program

The USENIX Association sponsors an Educational Outreach Program for faculty members and staff of computer science departments around the world. Outreach liaisons provide their students access to USENIX publications, upcoming events as well as conference scholarships for full-time students. For more information, please contact Diane DeMartini <*diane@usenix.org*>.

A $500 cash prize is awarded for the Best Student Paper at the annual Technical Conference. (Students are eligible also for Best Paper and other awards.)

Membership in USENIX for full-time Students is only $25. As a member, your benefits include:

- $75 student conference registration fees (vs fees of $295 to $380 for as many as ten technical conferences and symposia each year)
- $50 tutorial fees for students only
- Free subscription to *;login:*, bimonthly newsletter with articles, community news, calendar of events, book reviews, Standards Activities Reports, Systems Administrators Guild News, and more
- Free subscription to *Computing Systems*, refereed technical quarterly published with The MIT Press
- Access to papers from USENIX Conference and Symposia, starting in 1994, via the USENIX Online Library on the World Wide Web <*http://www.usenix.org*>
- Discounts on conference/symposia proceedings, other technical publications, and books from the new series on advanced computing published with MIT Press
- Discount on BSDI, Inc. products
- Eligibility to join SAGE, the Systems Administrators Guild

USENIX

January 22-26, 1996
San Diego Marriott Hotel and Marina
San Diego, California

technical conference 1996

Announcement and Preliminary Call for Papers

Dates for Refereed Paper Submissions

Extended Abstracts or Manuscripts Due: *July 18, 1995*
Notification to Authors: *August 31, 1995*
Camera-ready Papers Due: *November 14, 1995*

Conference Schedule Overview

Tutorials: *January 22-23, 1996*
Technical Sessions: *January 24-26, 1996*
Birds-of-a-Feather Sessions: *January 23-25, 1996*
USENIX Reception: *January 24, 1996*
Vendor Display: *January 24-25, 1996*

Program Committee

Robert Gray, *US WEST, Program Chair*
Miche Baker-Harvey, *Digital Equipment Corporation*
David Black, *Open Software Foundation*
Matt Blaze, *AT&T Bell Laboratories*
John Kohl, *Atria Software, Inc.*
Darrell Long, *University of California at Santa Cruz*
John Ousterhout, *Sun Microsystems*
Christopher Small, *Harvard University*
Jonathan Smith, *University of Pennsylvania*
Carl Staelin, *Hewlett-Packard*
Brent Welch, *Xerox PARC*

Call for Submissions

The 1996 USENIX Technical Conference Program Committee seeks original and innovative papers about the applications, architecture, implementation, and performance of modern computing systems. As at all USENIX conferences, papers that analyze problem areas and draw important conclusions from practical experience are especially welcome. Some particularly interesting hot application topics are:

Privacy and cryptography
Compression applications
User interface toolkits
Nomadic and wireless computing
Networks and distributed computing
Security

A major focus of this conference is operating systems practice and experience. We seek papers describing original work and results in the design, implementation, and use of modern operating systems. Submissions describing extensions or modifications of complete and widely used operating systems are particularly encouraged in addition to those describing research systems or prototypes. Topics of interest in this area include but are not limited to:

OS structure and organization
Performance and optimization
OS support for real time & multimedia
OS support for embedded systems
OS interaction with HW architecture
Microkernel internals, servers, and applications

Note that the USENIX conference, like most conferences and journals, requires that papers not be submitted simultaneously to more than one conference or publication, and that submitted papers not be previously or subsequently published elsewhere. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings.

Cash Prizes

Cash prizes will be awarded for the best paper at the conference and the best paper by a full-time student.

How to Submit a Refereed Paper

It is important that you contact the USENIX Association office to receive detailed guidelines for submitting a paper to the refereed track of the technical sessions. Please telephone 510 528 8649 or send email to: *usenix96authors@usenix.org.*

In addition, specific questions about submissions to the USENIX 1996 Technical Conference may be made to the program chair via email at: *gray@usenix.org.*

The program committee will review full papers or extended abstracts. An extended abstract should be five manuscript pages (single-sided) or fewer in length. It should represent the paper in "short form." If the full paper has been completed, it may be submitted instead of an extended abstract. Full papers should be limited to 12 single-spaced pages.

Include references to establish that you are familiar with related work, and, where possible, provide detailed performance data to establish that you have a working implementation and measurement tools.

Where to Send Submissions

Please send one copy of an extended abstract to the program chair via one of the following methods. All submissions will be acknowledged.

Preferred Method: email (PostScript or ASCII) to *usenix96papers@usenix.org*

Alternate Method: postal delivery to
Robert Gray
U S WEST Technologies
4001 Discovery Drive, Suite 280
Boulder, CO 80303
Phone: 303 541 6014

The authors must also submit via email to *usenix96papers@usenix.org* the following information:

1. The title of the manuscript and the names of the authors.
2. The name of one author who will serve as a contact, an email address, day and evening phone numbers, postal mail address, and a fax number, if available.
3. An indication of which, if any, of the authors are full-time students.
4. A short abstract of the paper (75-150 words).

Tutorials

On Monday and Tuesday, you may attend intensive, immediately practical tutorials on topics essential to the use, development, and administration of UNIX and UNIX-like operating systems, windowing systems, networks, advanced programming languages, and related technologies. The USENIX Association's well-respected program offers you both introductory and advanced courses, presented by skilled teachers who are hands-on experts in their topic areas.

USENIX will offer two days of tutorials covering topics such as:

System administration
Systems and network security
Distributed computing: DCE, DFS, RPC, CORBA
Kernel internals: SVR4, Chorus, Windows NT
Systems programming tools and program development
Portability and interoperability
Client-server application design and development
Sendmail, DNS, and other networking issues
GUI technologies and builders

To provide the best possible tutorial slate, USENIX constantly solicits proposals for new tutorials. If you are interested in presenting a tutorial, contact the Tutorial Coordinator:

Daniel V. Klein
Phone: 412 421 2332
Email: *dvk@usenix.org*

Invited Talks

An Invited Talks track complements the Refereed Paper track. These talks by invited experts provide introductory and advanced information about a variety of interesting topics, such as using standard UNIX tools, tackling system administration difficulties, or employing specialized applications. Submitted Notes from the Invited Talks are published and distributed free to conference technical sessions attendees. This track also includes panel presentations and selections from the best presentations offered at 1995 USENIX conferences and symposia.

The Invited Talks coordinators welcome suggestions for topics and request proposals for particular Talks. In your proposal, state the main focus, include a brief outline, and be sure to emphasize why your topic is of general interest to our community. Please submit via email to: *ITusenix@usenix.org.*

Invited Talks Coordinators
Mary Baker, *Stanford University*
Ed Gould, *Digital Equipment Corporation*

Work-in-Progress Reports

Do you have interesting work you would like to share, or a cool idea that is not ready to be published? Work-in-Progress Reports are for you! Work-in-Progress Reports, scheduled during the technical sessions, introduce interesting new or ongoing work. The USENIX audience provides valuable discussion and feedback. We are particularly interested in presentation of student work. To schedule your report, contact Peg Schafer via email at *wips@usenix.org.*

Birds-of-a-Feather Sessions

The always popular Birds-of-a-Feather sessions (BOFs) are very informal gatherings of persons interested in a particular topic. BOFs often feature presentations or demonstrations followed by discussion, announcements, and the sharing of strategies. BOFs are offered Tuesday, Wednesday, and Thursday evenings of the conference. BOFs may be scheduled at the conference or in advance by telephoning the USENIX Conference office at 714 588 8649 or via email to: *conference@usenix.org.*

Vendor Display

In the USENIX Vendor Display, emphasis is on serious answers from technically savvy vendor representatives. Vendors will demonstrate the technical innovations which distinguish their products. Here, in a relaxed environment, conference attendees can ask questions and discuss how something will work with what they already have. Plus, you can review the newest releases from technical publishers.

Vendors: This is an exceptional opportunity for receiving feedback from USENIX's technically astute conference attendees. If your company would like to display its products and services, please contact:
Zanna Knight. Telephone: 510 528 8649
Fax: 510 548 5738 Email: *display@usenix.org*

Conference Program and Registration Information

Materials containing all details of the technical sessions and tutorial program, conference registration, hotel and airfare discounts, and reservation information will be available at the end of September 1995. If you wish to receive the registration materials, please contact:

USENIX Conference Office
22672 Lambert St., Suite 613
Lake Forest, CA USA 92630
Phone: 714 588 8649
Fax: 714 588 9706
Email: *conference@usenix.org*
WWW URL: *http://www.usenix.org.*

Or send email to our mailserver at: *info@usenix.org.* Your message should contain the line: *send catalog.* A catalog will be returned to you.

The USENIX Association

Since 1975 the USENIX Association has brought together the community of engineers, scientists, and technicians working on the cutting edge of the computing world. The USENIX technical conferences and symposia have become the essential meeting grounds for the presentation and discussion of the most advanced information on new developments in all aspects of computing systems.

The USENIX Association and its members are dedicated to:

- problem-solving with a practical bias,
- fostering innovation and research that works,
- communicating rapidly the results of both research and innovation, and
- providing a neutral forum for the exercise of critical thought and the airing of technical issues.

SAGE, the System Administrators Guild, a Special Technical Group within the USENIX Association, is dedicated to the recognition and advancement of system administration as a profession.

A UNIQUE OFFER ON THE BEST IN UNIX FOR USENIX MEMBERS

20% DISCOUNT FROM McGRAW-HILL

□ **THE INTERNET GUIDE FOR NEW USERS**
D. Dern
hardcover, 016510-6, $40.00,
MEMBER PRICE **$32.00**
paperback, 016511-4, $27.95,
MEMBER PRICE **$22.36**

□ **INTERNET FOR EVERYONE**
R. Wiggins
hardcover, 067018-8, $29.95,
MEMBER PRICE **$23.96**
paperback, 067019-6, $45.00,
MEMBER PRICE **$36.00**

□ **THE ESSENTIAL INTERNET INFORMATION GUIDE**
J. Manger
707905-1, paperback, $27.95,
MEMBER PRICE **$22.36**

□ **THE INFORMATION BROKERS HANDBOOK, Second Edition**
S. Rugge
911878-x, paperback, $34.95,
MEMBER PRICE **$27.96**
Available December l994

□ **SAA AND UNIX: IBM's Open System Strategy**
M. Killen
034607-0, $40.00,
MEMBER PRICE **$32.00**

□ **A STUDENT'S GUIDE TO UNIX**
H. Hahn
025511-3, paperback, $28.00,
MEMBER PRICE **$22.40**

□ **UNIX DEVELOPER'S TOOL KIT**
K. Leininger
911836-4, $65.00,
MEMBER PRICE **$52.00**

□ **UNIX SECURITY: A Practical Tutorial**
N. Arnold
002560-6, $24.95,
MEMBER PRICE **$19.96**

□ **THE UNIX AUDIT: Using UNIX to Audit UNIX**
M. Grottola
025127-4, $32.95,
MEMBER PRICE **$26.36**

□ **UNIX: A Database Approach**
S. Das
015745-6, $29.95,
MEMBER PRICE **$23.96**
Available November l994

I am a member of USENIX Association. Please send me the books I have indicated at the member special rate. I have added $3.00 postage and handling for the first book ordered, $1.00 for each additional book, plus my local sales tax.

Check or money order is enclosed—payable to McGraw-Hill, Inc.
Charge my ❑ Visa ❑ Mastercard
❑ Discover ❑ Amex

Account #____________________

Expiration Date____________________

Signature____________________

USENIX Membership # ________________

Bill & Ship To:

Name____________________

Street____________________

City, State, Zip____________________

Daytime Phone #____________________

03US002

Send or Fax Orders to:
McGraw-Hill, Inc.
Attn: Rosa Perez
11 West 19th Street—4th Floor
New York, New York 10011
Fax: 212-337-4092

If I am not completely satisfied, I will return the book(s) within 15 days for a full refund or credit. Satisfaction unconditionally guaranteed. Prices subject to change without notics. We can only accept orders within the continental USA.

Computer Publications from John Wiley & Sons, Inc.

USENIX members receive
a 15% discount

The Internet Navigator, 2ed
Paul Gilster
1-05260-4 $24.95 **member price: $21.20**
# of copies: ________

Advanced Topics in UNIX
Ronald Leach
1-03663-3 $24.95 **member price: $21.20**
# of copies: ________

Introduction to Client Server Systems
Paul Renaud
1-57774-X $34.95 **member price: $29.70**
# of copies: ________

Portable UNIX
Douglas Topham
1-57926-2 $14.95 **member price: $12.71**
# of copies: ________

UNIX, Self-Teaching Guide
George Leach
1-57924-6 $19.95 **member price: $16.95**
# of copies: ________

Object Oriented Programming with Turbo C++
Keith Weiskamp
1-52466-2 $24.95 **member price: $21.20**
# of copies: ________

Obfuscated C and Other Mysteries
Don Libes
1-57805-3 $39.95 **member price: $33.96**
# of copies: ________

Finding It On the Internet
Paul Gilster
1-03857-1 $19.95 **member price $16.95**
# of copies: ________

Internationalization: Developing Software for Global Markets 1-07661-9 ***(pub. date: 1/95)***
Tuoc Luong $29.95 **member price: $25.45**
# of copies: ________

Adventures in UNIX Network Applications Programming
Bill Rieken
1-52858-7 $39.95 **member price: $33.96**
# of copies: ________

UNIX Shell Programming, 3e
Lowell Jay Arthur
1-59941-7 $29.95 **member price: $25.45**
# of copies: ________

The UNIX Command Reference Guide
Kaare Christian
1-85580-4 $32.95 **member price: $28.01**
# of copies: ________

Berkeley UNIX: A Simple & Comprehensive Guide
James Wilson
1-61582-X $40.95 **member price: $34.80**
# of copies: ________

□ Payment enclosed, plus sales tax
□ VISA □ Mastercard
□ American Express
Card No. ________
Name ________
Firm ________
Address________
City/State/Zip________
Signature________
(order invalid unless signed

Please send all orders to:

John Wiley & Sons, Inc.
Attn: Karen Cooper, Special Sales
605 Third Avenue
New York, NY 10158
Phone: (212) 850-6789
Fax: (212) 850-6142

What is Open Systems and where can you find the information you need to keep up in today's competitive open systems environment?

Find out by joining UniForum: The International Association of Open Systems Professionals.

The Advantages of Membership:

Your annual membership is a resource treasure chest. Among the many valuable benefits you'll receive are:

- *UniForum Monthly*: The Magazine for Open Systems Professionals
- *UniNews* - the authoritative voice of UniForum - now on-line
- *The Open Systems Products Directory* - comprehensive, accurate, indispensable
- Reduced fees at all UniForum conferences and seminars
- Technical and Standards publications: timely, reliable and useful
- Discounts on many outstanding industry publications
- Discounts on leading commercial hardware and software products
- Discounts on courses offered by leading training organizations
- Discounts on exclusive new shareware selections
- Services such as discounts on car rentals, travel and mail list rentals
- Eligibility to serve on UniForum committees and the Board of Directors

UNIFORUM MEMBERSHIP APPLICATION FORM *(please print)*

Name ______________________________ MR/MS

Title ______________________________

Company ______________________________

Address ______________________________

Mail Stop ______________________________

City __________ State __________ Zip __________

Country __________ Telephone (______) __________

E-mail (where we will send UniNews) __________ Fax (______) __________

General Membership** $125

Includes *UniForum Monthly* magazine, *UniNews* electronic newsletter, *Open Systems Products Directory,* technical publications, recruitment advertising service, and discounts on: UniForum conference registration and shareware CD-ROMs, purchases of software and hardware, educational seminars and special classes, additional UniForum publications and other industry publications, computer books, CD-ROMs and training services.

** Overseas postage:
air $100 or surface $60 (no additional postage necessary for Canada, Mexico or Puerto Rico) $__________

TOTAL AMOUNT ENCLOSED $__________

Method of Payment:

Select one: ❑ Check payable to UniForum *❑ Money Order In US Dollars
❑ MasterCard ❑ Visa ❑ American Express

Credit card number __________ Expiration date __________

Signature ______________________________

*Payments must be included in U.S. dollars. All checks must be drawn on a U.S. bank. Credit card orders can be accepted by telephone.
**Overseas surface delivery takes up to six weeks.

Send application and payment to: UniForum, 2901 Tasman Drive, Suite 205, Santa Clara, CA 95054-1100
Tel: (408) 986-8840 (800) 255-5620 Fax: (408) 986-1645

Prices subject to change without notice. Contact UniForum for discounts and shipping and handling charges on bulk orders.

UniForum.
The International Association of Open Systems Professionals

There's Never Been a Better Time for an Outstanding Value

As the unstoppable move to open systems gathers even more momentum you need a strong and independent association that can help you achieve your professional goals. An association that cal magnify your voice, which can educate, inform and inspire. You need and deserve an association that provides value for you money. In short you need UniForum. With a value this good the real question migh be, "Can you afford tc be without it?" Join today!

SPECIAL ANNOUNCEMENT

NEW EDUCATIONAL PROGRAMS FROM UNIFORUM

UniForum
The International Association of Open Systems Professionals

The UniForum Professional Training Series

Come and Learn! UniForum is pleased to announce its new training series — the first series consists of six comprehensive seminars in a modular format — designed for the novice, intermediate and advanced users of Open Systems — and they're coming to a city near you!

JUNE 12-16, 1995
AT THE HYATT HOTEL, BURLINGAME, CA
SAN FRANCISCO BAY AREA

◆ ◆ ◆

NOVEMBER '95 IN DALLAS, TEXAS

Compelling Reasons for You to Attend!

1 It's What You Need to Stay on Top! You're being asked to learn more to do more. You're being challenged to take on new responsibilities that can mean the difference between success on the job — or finding a new one! UniForum Professional Training Programs are designed with your career needs in mind. You'll get training and education you can apply to the job right now.

The Choice is Yours!

- Understanding the Internet & Internet Providers
- Security on the Internet & the World-Wide Web
- Designing & Building Your Company's World-Wide Web Server
- Enterprise Systems Administration/Management
- Enterprise Network Administration/Management
- Performance Management & Tuning Part I & II

2 The Best Instructors and a Modular Curriculum. You'll capture in one to five days what could take years to learn on the job. The choice is yours: attend one, two, three, four or all five days.

3 You'll Receive a Certificate of Advanced Achievement. UniForum is committed to promoting education about the benefits of opens systems and related hardware, software, applications and standards. UniForum is a recognized leader in Open Systems education and training.

4 UniForum Professional Training Series Guarantee. We guarantee we won't waste your time. The UniForum Professional Training Series will provide relevant information you need. If you're not completely satisfied with the value and benefits gained from attending this series, we'll refund your tuition in full or arrange for you to attend another UniForum Training Program at no additional fee.

To find out more about the UniForum Professional Training Series call: UniForum 1-800-255-5620, ext. 34 or send your email to: conferences@uniforum.org

The UniForum Security Track for Managers

The Past, Present and The Future of Information Security

JUNE 5-7, 1995
AT THE 5TH USENIX UNIX SECURITY SYMPOSIUM
MARRIOTT HOTEL, SALT LAKE CITY, UT

Sponsored by the USENIX Association, in cooperation with:
UniForum Association,
The Computer Emergency Response Team (CERT),
and IFIP WG 11.14

Overview: Information has become a critical business asset, crucial to the success of every business unit. Businesses in the future will rise and fall based in large measure on how well they manage and control their business information resources, and how effectively they use the resources and services of the (inter) national information infrastructure. Paramount in this information age of the future is the assurance of security. This dynamic two-day program will provide attendees with a comprehensive overview of computer security as it relates to Open Systems from a manager's perspective. The first day is designed for managers just getting into the business. It will provide information about what security is, it's components, and the risks to information today's environment from hackers, viruses, networking, and the World-Wide Web. The second day will provide attendees with more in-depth information regarding the necessary components involved with computer security. A host of security experts will lead four intensive sessions addressing: confidentiality, integrity, implementation issues and disaster planning and practice.

Who Should Attend: UNIX Site Managers, Directors/Managers MIS, Computer Security Managers, Database Managers, Systems Analysts/ Administrators/ Integrators, and anyone with an interest in addressing computer security within their business environment.

DAY ONE: *Information Security: Past, Present and Future*
Chair: Michael Weidner, Founder & VP of Arca Systems

DAY TWO: *The Four Basic Components within a Secure Computer Environment*

- **Session 1: Integrity**
  Chair: **Dr. Gene Schultz**, Deputy Program Manager, SRI
- **Session 2: Implementation Issues**
  Chair: **Katherine Fithen**, Technical Coordinator, CERT
- **Session 3: Disaster Recovery Planning & Practices**
  Chair: **Dr. Gene Spafford**, Director of Coast Laboratory, Purdue University
- **Session 4: Confidentiality**
  Chair: **Steve Walker**, Founder and President, Trusted Information Services

To find out more about the UniForum Security Track call: UniForum 1-800-255-5620, ext. 34 or send your email to: conferences@uniforum.org

ORDER FORM

CONFERENCE & WORKSHOP PROCEEDINGS

USENIX

Qty	*Proceedings*		*Member Price*	*Non-Member• Price*	*Subtotal*	*Overseas Postage*	*Total*
WINTER/SUMMER CONFERENCES							
____	New Orleans	Winter '95	30	39	$____	20	$____
____	Boston	Summer '94	25	33	$____	18	$____
____	San Francisco	Winter '94	30	39	$____	20	$____
____	Cincinnati	Summer '93	25	33	$____	18	$____
____	San Diego	Winter '93	33	40	$____	25	$____
____	San Antonio	Summer '92	23	30	$____	14	$____
____	San Francisco	Winter '92	30	39	$____	22	$____
____	Nashville	Summer '91	32	38	$____	22	$____
____	Dallas	Winter '91	28	34	$____	18	$____
____	Anaheim	Summer '90	22	22	$____	15	$____
____	Washington, DC	Winter '90	25	25	$____	15	$____
____	Baltimore	Summer '89	20	20	$____	15	$____
____	San Diego	Winter '89	30	30	$____	20	$____
____	San Francisco	Summer '88	29	29	$____	20	$____
____	Dallas	Winter '88	26	26	$____	15	$____
____	Phoenix	Summer '87	35	35	$____	20	$____
____	Washington, DC	Winter '87	10	10	$____	15	$____
____	Atlanta	Summer '86	37	37	$____	20	$____
____	Denver	Winter '86	25	25	$____	15	$____
____	Portland	Summer '85	45	45	$____	25	$____
____	Dallas	Winter '85	15	15	$____	10	$____
____	Salt Lake City	Summer '84	29	29	$____	20	$____
____	Washington, DC	Winter '84	25	25	$____	15	$____
____	Toronto	Summer '83	32	32	$____	20	$____
____	San Diego	Winter '83	28	28	$____	15	$____
LARGE INSTALLATION SYSTEMS ADMINISTRATION							
____	LISA VIII	Sept. '94	22	29	$____	10	$____
____	LISA VII	Nov. '93	25	33	$____	12	$____
____	LISA VI	Oct. '92	23	30	$____	12	$____
____	LISA V	Sept. '91	20	23	$____	11	$____
____	LISA IV	Oct. '90	15	18	$____	8	$____
____	LISA III	Sept. '89	13	13	$____	9	$____
____	LISA II	Nov. '88	8	8	$____	5	$____
____	LISA I	April '87	4	4	$____	5	$____
C++							
____	C++ Conference	April '94	24	28	$____	20	$____
____	C++ Conference	Aug. '92	30	39	$____	20	$____
____	C++ Conference	April '91	22	26	$____	11	$____
____	C++ Conference	April '90	28	28	$____	18	$____
____	C++ Conference	Oct. '88	30	30	$____	20	$____
____	C++ Workshop	Nov. '87	30	30	$____	20	$____
SECURITY							
____	UNIX Security IV	Oct. '93	15	20	$____	9	$____
____	UNIX Security III	Sept. '92	30	39	$____	11	$____
____	UNIX Security II	Aug. '90	13	16	$____	8	$____
____	UNIX Security	Aug. '88	7	7	$____	5	$____
MACH							
____	Mach Symposium III	April '93	30	39	$____	18	$____
____	Mach Symposium	Nov. '91	24	28	$____	14	$____
____	Mach Workshop	Oct. '90	17	20	$____	9	$____

Qty	*Proceedings*		*Member Price*	*Non-Member Price*	*Subtotal*	*Overseas Postage*	*Total*
	DISTRIBUTED & MULTIPROCESSOR SYSTEMS (SEDMS)						
____	SEDMS IV	Sept. '93	24	32	$________	14	$________
____	SEDMS III	Mar. '92	30	36	$________	20	$________
____	SEDMS II	Mar. '91	30	36	$________	20	$________
____	SEDMS	Oct. '89	30	30	$________	20	$________
	MICROKERNELS & OTHER KERNEL ARCH.						
____	Microkernels & Other Kernel... II	Sept. '93	15	20	$________	9	$________
____	Microkernels & Other Kernel... I	Apr. '92	30	39	$________	20	$________
	GRAPHICS						
____	Graphics Workshop V	Nov. '89	18	18	$________	10	$________
____	Graphics IV	Oct. '87	10	10	$________	10	$________
____	Graphics III	Nov. '86	10	10	$________	5	$________
____	Graphics II	Dec. '85	7	7	$________	5	$________
	OTHER WORKSHOPS/SYMPOSIA						
____	Mobile Computing	Apr. '95	18	24	$________	9	$________
____	OS Design and Implementation	Nov. '94	20	27	$________	11	$________
____	Very High Level Languages	Oct. '94	23	30	$________	10	$________
____	High-Speed Networking	Aug. '94	15	20	$________	9	$________
____	UNIX Applications Development	April '94	15	20	$________	9	$________
____	Mobile Computing	Aug. '93	15	20	$________	8	$________
____	File Systems	May '92	15	20	$________	9	$________
____	UNIX Transaction Processing	May '89	12	12	$________	8	$________
____	Software Management	Apr. '89	20	20	$________	15	$________
____	UNIX & Supercomputers	Sept. '88	20	20	$________	10	$________
	SAGE Short Topics System Administration Series Short Topics System Administration Series						
____	#1: Job Descriptions for System Administrators		5	7.50	$________	3.50	$________

Discounts are available for bulk orders. Please inquire.

Total price of Proceedings ________**
Calif. residents add sales tax ________
Total overseas postage ________
Total enclosed ________

****If you are paying member price, please include member's name and/or membership number**________________________________

PAYMENT OPTIONS*

____ Check enclosed- payable to USENIX Association Account # ______________________ Exp.Date __________

____ Charge my: ____ VISA ____ MC Signature ________________________________

____ Purchase order enclosed

* Outside the USA? Please make your payment in US currency by one of the following:
- Check - issued by a local branch of a US Bank
- Charge (VISA, MasterCard, or foreign equivalent)
- International postal money order

• If you are not a member and wish to receive our membership information packet, please check this box. □

Shipping Information

Ship to: ________________________________
____
____
____

Please allow 2 – 3 weeks for delivery. Overseas orders are shipped via air printed matter. Please mail or fax this order form with payment to:

USENIX Association • 2560 Ninth Street • Suite 215 • Berkeley, CA • 94710 • TEL 510-528-8649 • FAX 510-548-5738

LOCAL USER GROUPS

The Association will support local user groups by doing a mailing to assist in the formation of a new group and publishing information on local groups in *;login:*. At least one member of the group must be a current member of the Association. Send additions and corrections to: <*login@usenix.org*>.

California

Fresno:

The Central California UNIX Users Group has a WWW contact page to which members may post questions or information. For connection information:

- Steve Mitchell
  209 278-5675
  <*http://warpig.cati.csufresno.edu/ccuug/ccuug.html*>

Orange County:

Meets the 2nd Monday of each month

- UNIX Users Association of Southern California
  Dave Close
  714 434-7359
  <*dhclose@alumni.caltech.edu*>
  New Horizons Computer Learning Center
  1231 E. Dyer Rd., Suite 140
  Santa Ana, CA 92705
  714 438-9440

Colorado

Boulder:

Meets monthly at different sites; for membership information and meeting schedule, send email to <*fruug-info@fruug.org*>.

- Front Range UNIX Users Group
  Lone Eagle Systems Inc.
  636 Arapahoe #10
  Boulder, CO 80302
  Steve Gaede
  303 444-9114
  <*gaede@fruug.org*>
  <*http://www.fruug.org/~fruug*>

Washington, D.C.

Meets 2nd Tuesday of each month.

- Washington Area UNIX Users Group
  10440 Shaker Drive, Suite 103
  Columbia, MD 21046
  Alan Fedder
  301 621-5500
  <*afedder@mcsp.com*>

Florida

Coral Springs:

- S. Shaw McQuinn
  305 344-8686
  8557 W. Sample Road
  Coral Springs, FL 33065

Melbourne:

Meets the 3rd Monday of every month.

- Space Coast UNIX User's Group
  Steve Lindsey
  407 242-4766
  <*lindsey@vnet.ibm.com*>

Orlando:

Meets the 3rd Thursday of each month.

- Central Florida UNIX Users Group
  Mikel Manitius
  407 384-4644
  <*mikel.manitius@east.sun.com*>

Western:

Meets 1st Thursday of each month.

- Florida West Coast UNIX Users Group
  Mike Delucia
  813 882-0770
  <*pfl@cftnet.com*>

Georgia

Atlanta:

Meets on the 1st Monday of each month in White Hall, Emory University.

- Atlanta UNIX Users Group
  P.O. Box 12241
  Atlanta, GA 30355-2241
  Mark Landry 404 365-8108

Kansas or Missouri

Meets on 2nd Tuesday of each month.

- Kansas City UNIX Users Group (KCUUG)
  P.O. Box 412622
  Kansas City, MO 64141
  816 891-1093
  <*richj@northcs.cps.com*>

Michigan

Detroit/Ann Arbor:

Meets on the 2nd Thursday of each month in Ann Arbor.

- Southeastern Michigan Sun Local Users Group and Nameless UNIX Users Group
  Steve Simmons' office: 313 769-4086
  home: 313 426-8981
  <*scs@lokkur.dexter.mi.us*>

Minnesota

Minneapolis/St. Paul:

Meets the 1st Wednesday of each month.

- UNIX Users of Minnesota
  17130 Jordan Court
  Lakeville, MN 55044
  Robert A. Monio
  612 220-2427
  <*pnessutt@dmshq.mn.org*>

Missouri

St. Louis:

- St. Louis UNIX Users Group
  P.O. Box 2182 St. Louis, MO 63158
  Terry Linhardt
  314 772-4762
  <*uunet!jgaltstl!terry*>

Nebraska

Omaha:

Meets monthly.

- /usr/group/nebraska
  P.O. Box 31012
  Omaha, NE 68132
  Phillip Allendorfer
  402 423-1400

New England

Northern:

Meets monthly at different sites.

- Peter Schmitt 603 646-2085
  Kiewit Computation Center
  Dartmouth College
  Hanover, NH 03755
  <peter.schmitt@dartmouth.edu>

New Mexico

Albuquerque:

ASIGUNIX meets every 3rd Wednesday of each month.

- Phil Hortz 505 275-0466.

New York

New York City:

Meets every other month in Manhattan.

- Unigroup of New York City
  G.P.O. Box 1931
  New York, NY 10116
  <uniboard@unigroup.org>
  J. P. Radley 212 877-0440

Oklahoma

Tulsa:

Meets 2nd Wednesday of each month.

- Tulsa UNIX Users Group, $USR
  Bill Hunt 918 494-4848
  <bhunt@tulsix.utulsa.edu>
  Mark Lawrence 918 749-7498
  <lawrence@tulsix.utulsa.edu>

Texas

Austin:

Meets 3rd Thursday of each month.

- Capital Area Central Texas UNIX Society (CACTUS)
  P.O. Box 9786
  Austin, TX 78766-9786
  Ronald S. Woan
  512 838-1254
  <president@cactus.org>
  <http://cactus.org>

Dallas/Fort Worth:

Meets the 1st Thursday of each month.

Dallas/Fort Worth UNIX Users Group
P.O. Box 867405
Plano, TX 75086
Evan Brown 214 519-3577
<evbrown@dsccc.com>

Houston:

Meets 3rd Tuesday of each month.

- Houston UNIX Users Group (Hounix) answering machine
  713 684-6590
  Jack Gilbert, President
  713 862-3637
  <jack@hounix.org>

Washington

Seattle:

Meets monthly.

- Seattle UNIX Group Membership
  Bill Campbell 206 947-5591
  6641 East Mercer
  Mercer Island, WA 98040-0820
  <bill@celestial.com>

Canada

Manitoba:

Meets 2nd Tuesday of each month.

- Manitoba UNIX User Group (MUUG) P.O. Box 130
  St. Boniface Winnipeg,
  MB R2H 3B4
  Bary Finch, President
  204 934-1690
  <info@muug.mb.ca>

Ottawa:

- The Ottawa Carleton UNIX Users Group
  David J. Blackwood
  613 957-9305
  <dave@revcan.ca>

Toronto:

- 143 Baronwood Court
  Brampton, Ontario
  Canada L6V 3H8
  Evan Leibovitch
  416 452-0504
  <evan@telly.on.ca>

Quebec:

Meets first Wednesday every 3rd month.

- Administrateurs de Système UNIX du Quebec (ASUQ)
  Université de Montreal,
  Dept. IRO
  C.P. 6128, Succ. Centre-Ville
  Montreal, Quebec, Canada,
  H3C 3J7
  514 343-7480
  <asuq@iro.umontreal.com>

System Administration Groups

Back Bay LISA (BBLISA)

New England forum covering all aspects of system and network administration, for large and small installations. Meets monthly, at MIT in Cambridge, MA.

For information, contact:

- J. R. Oldroyd 617 227-5635
  <jr@opal.com>
- Mailing list subscription:
  <bblisa-request@bblisa.org>
- Mailing list postings:
  <bblisa@bblisa.org>
- For current calendar of events:
  finger *<bblisa@finger.bblisa.org>*

Bay LISA (California)

Meets 3rd Thursday of each month at Synopsys in Mountain View, CA For more information, please contact: *<blw@baylisa.org>* or *<http://www.baylisa.org>*

$GROUPNAME (New Jersey)

$GROUPNAME is an organization in New Jersey formed to facilitate information exchange pertaining to the field of UNIX system administration. For more information, send "infogroupname" to *<majordomo@plts.org>*
Tom Limoncelli *<tal@big.att.com>*

New York Systems Administrators (NYSA)

Meets 2nd Monday of each month.

- *<nysa-request@esm.com>*
  914 472-3635 or 472-3635

North Carolina System Administrators Group

The North Carolina System Administrators Group meets on the 2nd Monday each month around the Research Triangle Park area.

- Amy Kreiling 919 962-1843
  <kreiling@cs.unc.edu>
- William E. Howell 919 941-4868
  <william_howell@glaxo.com>

CALENDAR OF EVENTS

This is a combined calendar of conferences, symposia, and standards meetings. If you have a event that you wish to publicize, please contact <*login@usenix.org*>.

* = events sponsored by the USENIX Association.

1995

June

5 - 7*	UNIX Security, Salt Lake City, UT
18-23	ACM SIGPLAN Conference, La Jolla, CA
26-29*	COOTS, Monterey, CA
27-29	WITI, Santa Clara, CA

July

6-11*	Tcl/Tk Workshop, Toronto, Canada
10-14	IEEE 1003
13-14*	Electronic Commerce, NY, NY
17-21	IETF, Stockholm, Sweden
17-21	NetWorld + Interop 95, Tokyo, Japan

August

1 - 4	4th IEEE Int'l Symposium on High Performance Distributed Computing, Pentagon City, VA
6-11	ACM SIGGRAPH, Los Angeles, CA
13-15	IEEE Hot Chips Symposium VII, Stanford, CA
14-18	Interex 95, Toronto, Canada

September

11-13	EurOpen, Security Symposium Stockholm, Sweden
12-14	GUUG Annual Conference, Wiesbaden, Germany
18-21	AUUG '95, Sydney, Australia
18-22*	LISA '95, Monterey, CA
20-23	IEEE 4th Intl. Conf. Computer Communications & Networks, Las Vegas, NV
19-21	UNIX Expo, New York City
25-29	NetWorld + Interop 95, Atlanta, GA

October

16-20	IEEE 1003
12-15	ACM SIGSOFT '95, Washington, DC
15-19	OOPSLA '95, Austin, TX
25-28	IEEE Parallel & Distributed Processing, San Antonio, TX
25-26	EurOpen, Publishing on Internet, Stockholm, Sweden

November

1 - 4	ROSE '95, Bucharest, Romania
2 - 8	DECUS, San Francisco, CA
6 -10	NetWorld + Interop 95, Paris
6 -10	IEEE Int'l. Conf. on Engineering of Complex Computer Systems, Ft. Lauderdale, FL

December

	JUS UNIX Fair, Tokyo, Japan
2 - 7	DECUS, San Francisco, CA
3 - 6	SOSP, Colorado
4 - 8	IETF, Dallas, TX
3- 8	ACM/IEEE-CSSupercomputing'95, San Diego, CA
11-14	4th World Wide Web Conference, Boston, MA

1996

January

22-26*	USENIX, San Diego, CA

February

14-16	UniForum, San Francisco, CA

March

	IETF

May

18-24	DECUS, Orlando, FL

August

4 - 8	Interex '96, San Diego, CA

September

30-O4*	LISA '96, Chicago, IL
	AUUG, Melbourne, Australia

October

2 - 4	ASPLOS (tentative)
7 -11	OOPSLA '96
8-10	UNIX Expo, New York City
29- N1	OSDI II, Seattle, WA

November

9 -14	DECUS, Anaheim, CA
18-22	ACM IEEE-CS Supercomputing '96, Pittsburgh, PA

ACM:
Association for Computing Machinery

AFUU:
Association of French UNIX Users

ASPLOS:
Architectural Support for Programming Languages and Operating Systems

AUUG:
Australian UNIX Systems Users Group

COOTS:
Conference on Object-Oriented Technologies

DECUS:
Digital Equipment Computer Users Society

EurOpen:
European Forum for Open Systems

GURU:
Romanian UNIX User Group

GUUG:
German UNIX Users Group

IEEE: Institute of Electrical and Electronics Engineers

IETF:
Internet Engineering Task Force

Interex:
International Association of Hewlett-Packard Computer Users

JUS:
Japan UNIX Society

LISA:
USENIX/SAGE Systems Administration Conference

OOPSLA:
Object-oriented Programming Systems, Languages and Applications

OSDI:
Symposium on Operating Systems Design & Implementation

ROSE:
Open Systems in Romania

SANS:
System Administration, Networking & Security

SIGPLAN:
ACM Special Interest Group on Programming Languages

SIGSOFT:
ACM Special Interest Group on Software Engineering

SOSP:
ACM Symposium on Operating Systems Principles

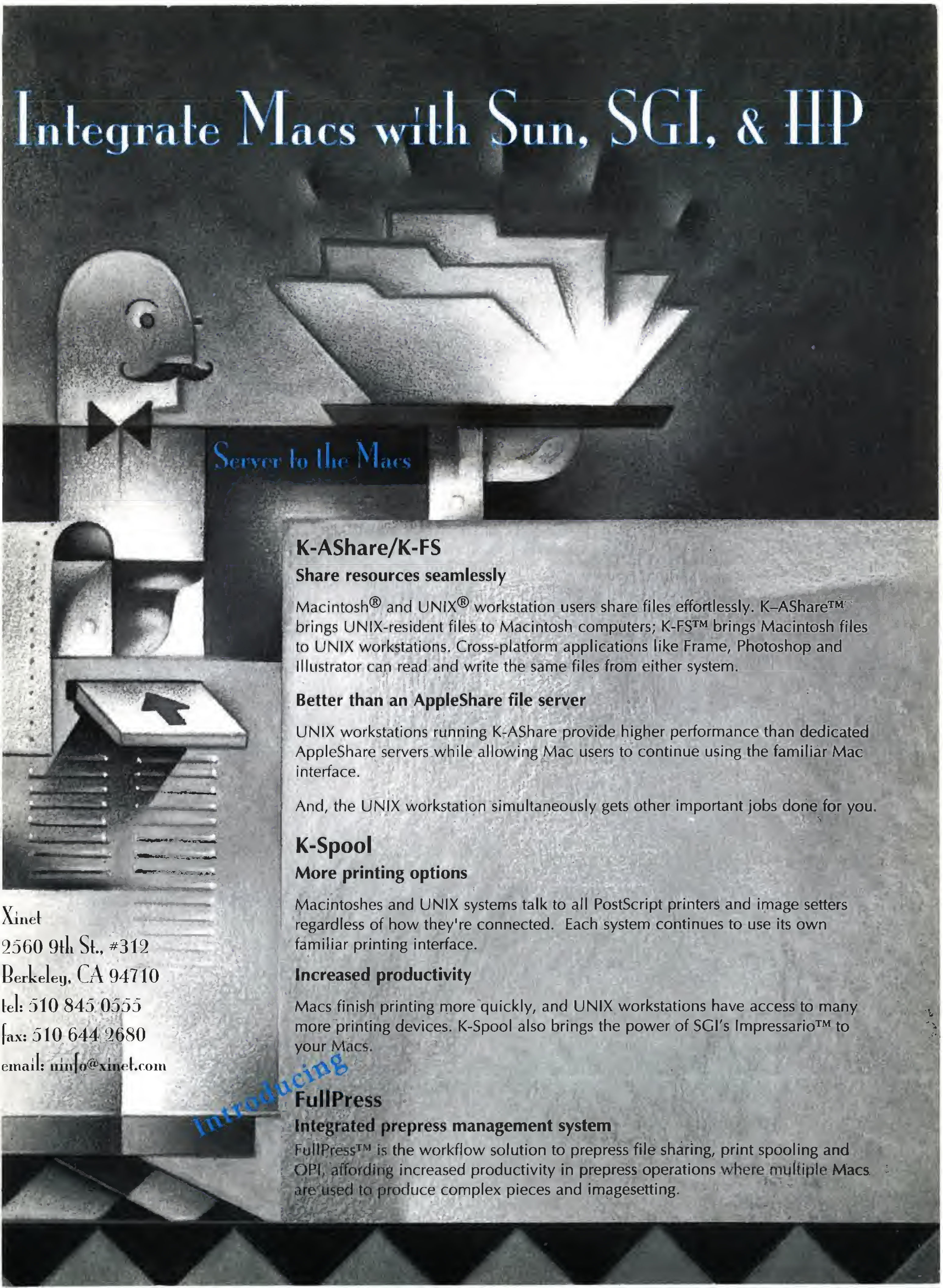
SUG:
Sun User Group

SUUG:
Society of Russia UNIX Users Group

UKUUG:
United Kingdom UNIX Systems Users Group

UniForum:
International Association of UNIX and Open Systems Professionals

WITI:
International Network of Women in Technology



Integrate Macs with Sun, SGI, & HP

Server to the Macs

K-AShare/K-FS

Share resources seamlessly

Macintosh® and UNIX® workstation users share files effortlessly. K–AShare™ brings UNIX-resident files to Macintosh computers; K-FS™ brings Macintosh files to UNIX workstations. Cross-platform applications like Frame, Photoshop and Illustrator can read and write the same files from either system.

Better than an AppleShare file server

UNIX workstations running K-AShare provide higher performance than dedicated AppleShare servers while allowing Mac users to continue using the familiar Mac interface.

And, the UNIX workstation simultaneously gets other important jobs done for you.

K-Spool

More printing options

Macintoshes and UNIX systems talk to all PostScript printers and image setters regardless of how they're connected. Each system continues to use its own familiar printing interface.

Increased productivity

Macs finish printing more quickly, and UNIX workstations have access to many more printing devices. K-Spool also brings the power of SGI's Impressario™ to your Macs.

Introducing

FullPress

Integrated prepress management system

FullPress™ is the workflow solution to prepress file sharing, print spooling and OPI, affording increased productivity in prepress operations where multiple Macs are used to produce complex pieces and imagesetting.

Xinet
2560 9th St., #312
Berkeley, CA 94710
tel: 510 845 0555
fax: 510 644 2680
email: ninfo@xinet.com

;login:

USENIX Association
2560 Ninth Street
Suite 215
Berkeley, CA 94710